

A wind-independent control strategy for autonomous sailboats based on Voronoi diagram

K. XIAO

*University of Minnesota,
Minneapolis, USA
E-mail: kaixiao1988@gmail.com
www.umn.edu*

J. SLIWKA* and L. JAULIN

*ENSTA-Bretagne, OSM, 2 rue Francois Verny,
Brest, 29200, FRANCE
* E-mail: jan.sliwka@ensta-bretagne.fr*

This paper proposes a new control strategy based on Voronoi diagram for autonomous sailboats. This strategy applies the method of Voronoi diagram to decompose the space and uses a lookup table to make the boat sail as planned. The contribution of this paper is that for the first time it proposes an approach to control autonomous sailboats without the data from wind sensors (with data from GPS and compass only). Since the sensor for detecting wind direction can break down at any time, this new strategy, independent of wind direction, is much more reliable. A simulation is proposed to illustrate the principle and the effectiveness of the approach. Two real experiments on buggy and sailboat are also performed for validation.

Keywords:
robotics, autonomous sailboat, control, wind-independent, Voronoi diagram.

1. Introduction

This paper puts up a new control strategy that is independent of wind sensor data for autonomous sailboats. It is motivated mainly by the *micro-transat challenge*, which focuses on the development of small autonomous sailboat robots that could go cross the Atlantic ocean. The challenge is quite meaningful since the unmanned sailboats powered by wind have a great potential application in future oceanographic observations and long term offshore operations (see,¹² and³).

Due to the harsh environment of the ocean, all components of such robots should be robust with respect to all situations (heavy weather, waves, salt water, low level of energy, long trip, . . .). Sensors on autonomous sailboats could be divided into two types: One type is *reliable sensors*, which could survive to all situations. Such sensors include the GPS (providing the position and speed of the boat with a good accuracy), the compass (measuring the north direction with a rather good accuracy in ocean), the gyrometers (giving the rotational speed of the boat) and accelerometers (measuring the roll and pitch of the robot). All these sensors are low energy consumers, can be enclosed inside a waterproof tank and can survive for years. The other type is *unreliable sensors*, which have a high probability to break down in case of heavy weather. Anemometer (device that is used for measuring wind speed), weather vane (which returns the direction of the wind) and dynamometer (which measures the forces on the sail or the rudder) are considered as unreliable. They are directly in contact with aggressive natural elements (wind, waves, salt) and can fail at any time. For simplicity, anemometers and weather vane are called wind sensors in the following passage.

Traditionally, to control the boat, it is necessary to get data from unreliable sensors in order to know where the wind comes from and how strong it is (see e.g.,^{4,56}). However, as said above, sensors for measuring the speed and direction of the wind are vulnerable and could not sustain for long periods⁽⁷⁾. Therefore, we intend to take a new control strategy that could operate without the knowledge of wind direction and speed.

In this paper, we come up with a new and simple control strategy that is wind-independent based on the Voronoi diagram. We have noticed that for every sailboat there exists a certain no-go-angle θ_{no} on the polar speed diagram⁽⁸⁾. It means that no matter where the wind comes from, if we provide the sailboat with m ($m \geq 2$) possible directions whose angular separations are greater than the no-go-angle, the sailboat could definitely follow one of them. Our new strategy is based on a lookup table which defines m ($m \geq 2$) possible directions for each zone of the sea. We have introduced the method of Voronoi diagram to implement the decomposition of the sea. As a result, we don't need any data about the wind direction and speed to control our sailboat.

Section 2 provides the basics of polar speed diagram, Voronoi diagram and presents the sailboat to be considered. Section 3 explains how our control strategy works. Section 4 provides a simulation of the strategy. Two real tests on buggy and sailboat are presented on Section 5 and Section 6



Fig. 1. sailboat with wind sensor

concludes the paper.

2. Preliminaries

2.1. Polar Speed Diagram

As we know, if the boat is pointed too close to the wind (unless they are backed), the sails will be luffing ("flapping") in the breeze and cannot generate any effective power, only making noise like a flag. Different boats, of course, have different performance characteristics. These characteristics depend on such variables as hull design, keel design, rigging, sails, etc.

The polar diagram of a sailboat is the set S of all pairs (θ, v) that can be reached by the boat, in a cruising mode. The area of direction that cannot be reached is called a no-go-zone. The size of the no-go zone (no-go-angle) will differ based on the performance characteristics of the particular sailboat. For example, racing sailboats can usually sail much closer to the wind (i.e., fewer degrees off the wind direction) than cruising yachts. This is known as "pointing higher."

However, for a given sailboat, the no-go-angle is certain and could be retrieved on its polar diagram. It means that the boat could definitely follow one of m ($m \geq 2$) predefined directions that have an included angle greater than its no-go-angle, whatever the wind direction is.

2.2. Voronoi Diagram

A Voronoi diagram is a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects in the space, e.g., by a discrete set of points. In the simplest case, we are given a set of points S in the plane, each point corresponding to one Voronoi site. Each point p has a Voronoi cell, $V(p)$ consisting of all points closer to p than to any other point. The segments of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest points.

In general, the set of all points closer to a point p of S than to any other point of S is the interior of a (in some cases unbounded) convex polytope, called the Dirichlet domain or Voronoi cell for p . The set of such polytopes tessellates the whole space, and is the Voronoi tessellation corresponding to the set S . If the dimension of the space is only 2, then it is easy to draw pictures of Voronoi tessellations, and in that case they are sometimes called Voronoi diagrams.

Given a set of $S \triangleq \{p_i \in \mathbb{R}^n\}$, the Voronoi cell associated with point p_i is the set

$$V(p_i) \triangleq \{x \mid d(x, p_i) \leq d(x, p_j), \forall i \neq j\}$$

and the Voronoi diagram of the set S is given by the union of all of the Voronoi cells.

Another way to define $V(p_i)$ is in terms of the intersection of halfplanes. Given two sites p_i and p_j , the set of points that are strictly closer to p_i than to p_j is just the open halfplane whose bounding line is the perpendicular bisector between p_i and p_j . Denote this halfplane $h(p_i, p_j)$. It is easy to see that a point q lies in $V(p_i)$ if and only if q lies within the intersection of $h(p_i, p_j)$ for all $j \neq i$. In other words,

$$V(p_i) \triangleq \bigcap h(p_i, p_j), (\forall j \neq i).$$

Since the intersection of halfplanes is a (possibly unbounded) convex polygon, it is easy to see that $V(p_i)$ is a (possibly unbounded) convex polygon.

There are a number of algorithms for computing Voronoi diagrams. Of course, there is a simple $O(n^2 \log n)$ time algorithm, which operates by computing $V(p_i)$ by intersecting all the bisector halfplanes $h(p_i, p_j), (\forall j \neq i)$. However, there are much more efficient ways like the sweepline algorithm developed by Steven Fortune⁹, which has $O(n \log n)$ worst-case running time.

Voronoi diagram has already been applied on Robotics. For example, it was employed on a path planning strategy for mobile robots on a two-

dimensional map, especially for avoidance of obstacles en route (see e.g.¹⁰).

2.3. Sailboat

In order to simulate our control strategy, we need a sailboat dynamics model to implement it. We have introduced here the following state equations based on those in.¹¹ According to our measurement and test, they are a good approximation of the dynamics of the sailboat represented on Figure 2

$$\begin{cases} \dot{x} = & v \cos \theta + p_1 a \cos \psi \\ \dot{y} = & v \sin \theta + p_1 a \sin \psi \\ \dot{\theta} = & \omega \\ \dot{v} = & \frac{f_s \sin \delta_s - f_r \sin u_1 - p_2 v}{p_9} \\ \dot{\omega} = & \frac{f_s (p_6 - p_7 \cos \delta_s) - p_8 f_r \cos u_1 - p_3 \omega + p_{11} \text{rand}(t)}{p_{10}} \\ f_s = & p_4 a \sin (\theta - \psi + \delta_s) \\ f_r = & p_5 v \sin u_1 \\ \gamma = & \cos (\theta - \psi) + \cos (u_2) \\ \delta_s = & \begin{cases} \pi - \theta + \psi & \text{if } \gamma \leq 0 \\ \text{sign} (\sin (\theta - \psi)) \cdot u_2 & \text{otherwise.} \end{cases} \end{cases}$$

The state vector $x = (x, y, \theta, v, \omega)^T$ is composed with

- the coordinates x, y of the inertial center G of the boat
- the orientation θ of the boat
- the tangential speed v of G
- the angular velocity ω of the boat around G

The internal variables are

- the thrust force f_s of wind on the sail
- the force f_r of water on the rudder
- the maximum possible angle of the sail δ_s at the current length of mainsheet (γ is to judge whether the sail is tight or not)

The parameters p_i, a, ψ are assumed to be known exactly. They are

- the drift coefficient p_1
- the tangential friction coefficient p_2 between the boat and water
- the angular friction coefficient p_3 between the boat and water
- the sail lift coefficient p_4 and the rudder lift coefficient p_5
- p_6, p_7, p_8 could easily be seen from Figure 2
- the mass of the boat p_9 and the angular inertia of the boat p_{10}

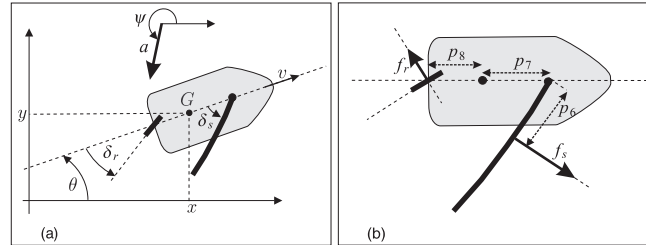


Fig. 2. Sailboat to be observed

- the speed a and direction ψ of the wind

The sailboat has two inputs: $u_1 = \delta_r$ is the angle between the rudder and the sailboat and u_2 is the current length of the mainsheet which limits δ_s . This model is similar to that described in,¹¹ except that here, (i) the control is not any more the sail angle, but the length of the mainsheet, which is more realistic, (ii) we added a random input on $\dot{\omega}$ that represents the random wave effect on the direction of the boat.

It should be noted that these state equations are introduced only for simulation and presentation purpose. They are not assumed as excellent models, just serving as an example. Any other dynamic models for sailboat could also be applied. Moreover, we simulate the behavior of the boat using Euler approximation.

3. Control Strategy

3.1. Algorithm

Our control algorithm is divided into two parts, one offline and the other online. The offline part is performed by the PC and needs human intervention while the online part is done by the microcontroller embedded in the sailboat.

3.1.1. The offline part

The offline part is mainly a planning step and needs human help. It takes time to draw the Voronoi diagram with our current technique and thus it is not suitable to be performed on the microcontroller. The PC could do the job and gives out the computed Voronoi diagram to the embedded microcontroller. The offline part is performed at the beginning of one certain

voyage.

step1: According to start and end points and obstacles en route, decompose the space by giving reference points that define the Voronoi diagram.

step2: Compute the Voronoi diagram.

step3: Assign each Voronoi site with m ($m \geq 2$) directions.

step4: Generate the C code for the microcontroller.

Remark 3.1. The angular separations between the m directions in one Voronoi site must be greater than the no-go-angle of the boat in order that the boat could follow at least one of them.

Remark 3.2. The effectiveness and efficiency of the strategy depends largely on the setting of reference points and directions. It needs to be done by an experienced sailor.

Remark 3.3. All the directions should ultimately lead to the target in some way.

Remark 3.4. To avoid obstacles some reference points should appear before them and directions associated with these sites should point outward from the obstacles.

3.1.2. *The online part*

The online part of the algorithm is the action step and performed by the embedded microcontroller. It takes the Voronoi diagram given by the PC and controls the action of the sailboat in real-time. The steps are as follows:

step1: Check which Voronoi site the sailboat is in.

step2: Choose the first direction assigned to the current site.

step3: Tune the sail so that the boat gets maximum speed on the target direction and control the rudder so that the boat goes on the target direction.

step4: If the boat fails to follow the current given direction, choose the next one of the current site.

step5: Go to *step3*.

step6: If it fails again, go to *step2*.

Remark 3.5. To determine whether the sailboat follows the desired direction, there could be many criteria using the sensor data about speed, direction or position of the boat. After comparison with other means, we

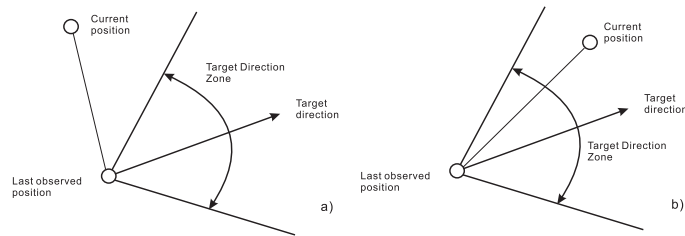


Fig. 3. a) the boat doesn't follow desired direction b) the boat follows target direction

chose the most well-proven way combining the data from GPS sensor and compass which are quite accurate in our scale of problem. The criteria is that if the actual direction of the boat during our observation time (the current position with respect to the last one observed) is within a certain zone around the target direction, then we assume the boat is successful in following it (shown on Figure 3).

3.2. *Tuning the Sail*

Because the wind direction is totally unknown, we cannot rely on any existing knowledge to control the angle of sail. As a result, we need to tune the sail to find the best sail angle so that the boat could fully utilize the power of the wind and follow the given direction. In our current strategy, we have two tuning functions. The first is to find the global optimal angle through a traversal technique, namely it tries every possible angle of sail and finds the best one. This function will be called when the given direction is changed, which means a totally different wind direction with respect to the boat will appear and therefore the best sail angle would be quite different from the last one. The second is to change the sail angle a little once at a time to try to find the local optimal angle. This function helps maintain the stability of the boat while searching for the best sail angle and avoids the high cost of time the first one needs.

3.3. *Controlling the Boat Direction*

In our control strategy, it is quite important for the boat to follow the given direction. For that purpose, a simple but efficient method for controlling the rudder should be applied, since the sailboat's direction is controlled mainly by the rudder. For most of the sailboats, the rudder is efficient enough to make the boat follow the target direction as long as it doesn't belong

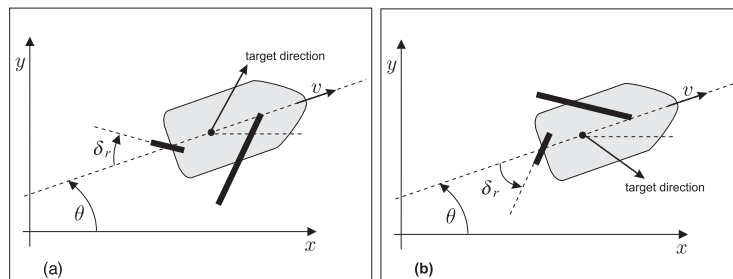


Fig. 4. rudder control strategy

to the no-go-zone of polar diagram. In our current strategy, the controller for the rudder is one kind of proportional (proportional to the difference between desired direction and current boat direction). Figure 4 shows how the controller works with different target directions.

4. Simulation

Applying the control strategy and the sailboat model on SCILAB, we conducted a simulation of the new control strategy. We chose the Brest Gulf as our simulation environment because we are familiar with its conditions. Again, it only serves to present our strategy and any other environment could be chosen for the test. We imported the boundary of Brest Gulf into the simulation program (the black lines represent the shoreline). According to our destination and desired path, we decomposed the map into 6 Voronoi sites (the red lines represent the boundary of Voronoi zones) and set the corresponding reference points (the red circles in the figure) along the desired path. To simplify the simulation without losing generality, we assigned each site with only two given directions (the two arrows associated with each reference point, with the darker one representing the first direction and the lighter one representing the second direction) that point to the destination in some way and make the boat avoid obstacles. Since the no-go-angle for most sailboats (including ours¹²) is less than 60 degrees, the included angle between the two directions of each Voronoi site should be greater than 60 degrees. The boat is represented by the black polygon. For simplicity, we used the straightforward algorithm previously described in Section 2.2:

$$V(p_i) \triangleq \cap h(p_i, p_j), (\forall j \neq i)$$

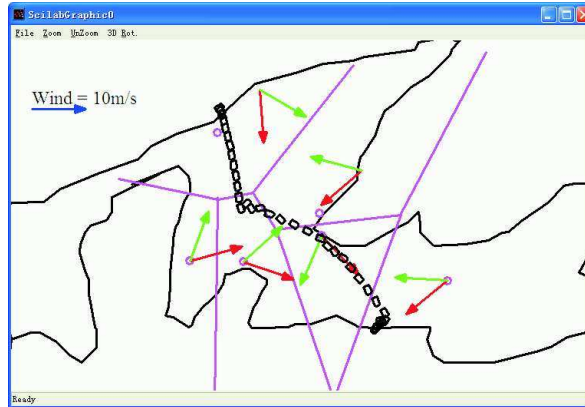


Fig. 5. simulation result

to compute the Voronoi diagram which costs $O(n^2 \log n)$ time. Due to its slowness, we computed the diagram once and stored it in the memory.

According to the real sailboat built by us¹², the parameter vector has been chosen as

$$\mathbf{p} = (0.1, 100, 500, 1500, 70, 1.1, 1.4, 2, 1000, 2000)^T.$$

The result is satisfactory: the boat goes from the starting point to the target without hitting obstacles in any given wind direction. One trajectory of the boat is shown on Figure 5.

The SCILAB code of the simulation as well as movies illustrating our simulation can be downloaded at

<http://www.ensieta.fr/jaulin/kai.html>

5. Testcase

5.1. *With Buggy*

In order to easily implement and see the result of our new control strategy, we tested first on a general ground robotic platform called buggy. There are some small differences between controlling the buggy and the sailboat since the buggy could follow any direction without difficulty. So we ignore the step3 and step5 of the online part that tune the sail. Instead, we just give the buggy a fixed speed. The steering servo which controls the direction of the buggy could be controlled as the rudder of the sailboat. And if we need to see what would happen when it fails to follow one direction, we



Fig. 6. buggy

could just move it in the wrong direction by hand. We chose a football field for test so that the GPS data would be quite accurate. At first, we intended the buggy to go around the field following some kind of spiral trajectory that directs the buggy to the center of the field. So we chose 4 reference points and decomposed the field into 4 sites. The sites and directions belonging to each site could be seen on the left half of Figure 7. The actual trajectory of the buggy is shown on the right half of the figure. The result was quite satisfactory, with the buggy following given direction quite well and it followed a spiral trajectory on the field as intended.

5.2. *With Sailboat*

After satisfactory results on buggy, we did a real test with our sailboat on Lake Tycolo near Brest, France. Further information about our sailboat could be seen in.¹² The only difference is that we unmounted the wind sensor (see Figure 8). Lake Tycolo is a small lake with good eyesight so that the behavior of the sailboat could be better watched during the test. For our test we intended the boat to travel from the bank of lake to the middle of it without hitting any boundary of the lake. According to our strategy, we decomposed the lake into 4 Voronoi sites. Again for simplicity, we assigned each site with only two directions whose angular separation is greater than no-go-angle (60 degrees for our testcase). Then, we implemented the on-line part of the control strategy described above on a PIC microcontroller embedded in our sailboat.

The result of the test is quite satisfactory. The boat could tune its sail so that it could find the best sail angle for the moment, although it does not have any information about the wind. The angle the boat finds by itself

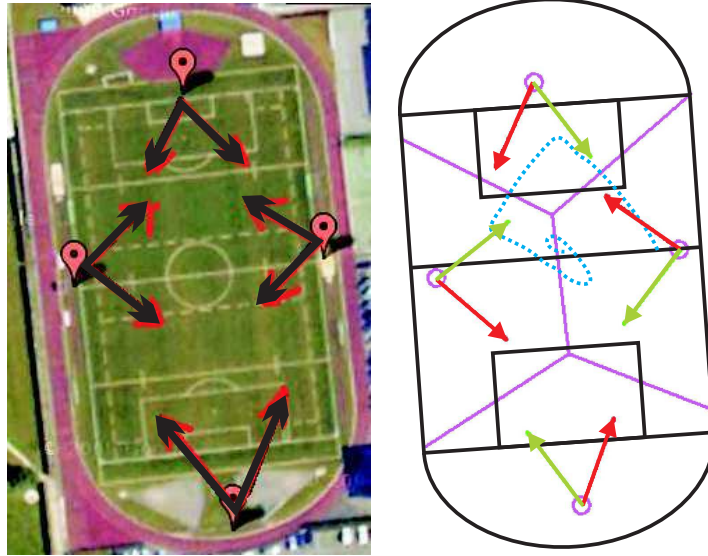


Fig. 7. test with the buggy



Fig. 8. sailboat without wind sensor

matches very well with human expectations. Moreover, the boat's trajectory is shown on Figure 9, which is almost as intended. The SCILAB code for

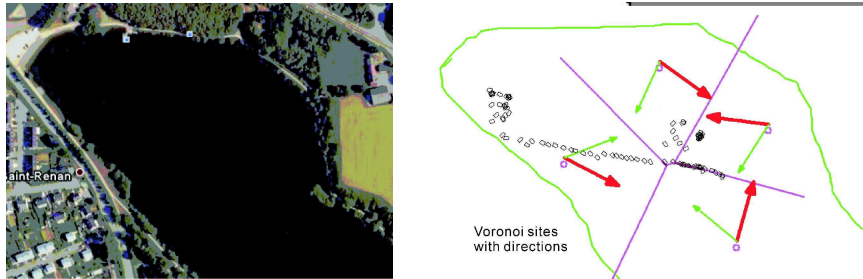


Fig. 9. test with the sailboat robot

the online part, the C code for the offline part of the strategy as well as the video showing our experiment can be downloaded at

<http://www.ensta-bretagne.fr/jaulin/kai.html>

6. Conclusion

In this paper, a new control strategy for autonomous sailboats based on Voronoi diagram has been presented with several advantages.

- The strategy's biggest advantage is that it is *wind-independent*, thus *much more reliable*. As shown above, since the boat could definitely follow one of the two given directions that ultimately point to the target, the boat will undoubtedly get to the target no matter where the wind comes from. Thus, the strategy would be much more reliable than most of others that depend highly on the data from wind sensors which are not so reliable especially in harsh environment. And due to the simplicity of the strategy, the new strategy would have little chance to fail.
- The strategy is *simple and easy to implement*. Due to the Voronoi diagram and the look-up table method, the strategy is quite simple and can be applied easily without putting a high demand on the hardware. Moreover, thanks to the simplicity of the strategy, the program costs few time to execute. In our real test for the sailboat, the computation time for one loop is at the millisecond level. With some advanced table look-up method and algorithms for generating Voronoi diagram, it could be even faster.

Although the strategy has been illustrated only on sailboats, it could have many other areas of application where some unreliable data are nec-

essary for a reliable control. Some potential application areas include space exploration robots, autonomous submarines, etc.

Some future work could help improve the strategy:

- Although the strategy works in our test, its effectiveness and efficiency depend largely on the quality of Voronoi diagram and the two given directions assigned to each Voronoi site. As a result, we currently need an experienced sailor to construct the Voronoi diagram and assign the two given directions. This would greatly limit its application. Future work could focus on finding some techniques to generate the Voronoi diagrams automatically. Many methods such as fuzzy logic, expert system and evolutionary algorithms could be turned for and there are a lot of interesting topics to discover.
- Another important improvement for our strategy is on the computation of Voronoi diagram. To compute the Voronoi diagram takes time. Our current solution is to compute it once and for all in the initialization stage of the program and store it in memory. But it has brought about another problem: in this way the Voronoi diagram could not be updated in real-time. Although the problem does not matter in most cases, it would limit the application of the strategy in some areas. The problem could be largely mitigated by applying some advanced algorithms for computing Voronoi diagram. Now we used a simple, reliable but time-consuming algorithm. There exist a great number of more efficient algorithms for computing Voronoi diagram and Delaunay triangulation. Applying them in our strategy could largely reduce the initialization time for the whole system and could even incorporate the computation in every loop so that the Voronoi diagram could be updated in real time.

References

1. P. F. Rynne and K. D. von Ellenrieder, *Marine Technology Society* **43**, 21 (2009).
2. N. A. Cruz and J. C. Alves, *Journal of the Oesterreichische Gesellschaft fuer Artificial Intelligence (Austrian Society for Artificial Inteligence)* **27**, 25 (2008).
3. Y. Briere, Iboat: An autonomous robot for long-term offshore operation, in *The 14th IEEE Mediterranean Electrotechnical Conference*, 2008.
4. C. Sauze and M. Neal, An autonomous sailing robot for ocean observation, in *in proceedings of TAROS 2006*,).

5. R. Stelzer and T. Proll, *Robotics and Autonomous Systems* **56**, 604 (2008).
6. M. Neal, *IEEE Journal of Oceanic Engineering* **31**, 462 (2006).
7. P. Herrero, L. Jaulin, J. Vehi and M. A. Sainz, *International Journal of Control Automation and Systems (to appear)* (2009).
8. J. Abril, J. Salmon and O. Calvo, *International Journal of Approximate Reasoning* **16** (1997).
9. S. J. Fortune, *Algorithmica* **2**, 153 (1987).
10. S. Garrido, L. Moreno, M. Abderrahim and F. Martin, Path planning for mobile robot navigation using voronoi diagram and fast marching, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing, China, 2006).
11. L. Jaulin, Modlisation et commande d'un bateau 'a voile, in *CIFA2004 (Confrence Internationale Francophone d'Automatique)*, In *CDROM*, (Douz (Tunisie), 2004).
12. J. Sliwka, P. Reilhac, R. Leloup, P. Crepier, H. D. Malet, P. Sittaramane, F. L. Bars, K. Roncin, B. Aizier and L. Jaulin, Autonomous robotic boat of ensieta, in *2nd International Robotic Sailing Conference*, (Matosinhos, Portugal, 2009).