



RESEARCH ASSISTANT INTERNSHIP REPORT

Smart monitoring and data analytics of coastal and marine environments using autonomous sailboats

Author

Agathe ARCHET
2nd year engineering student
Autonomous Robotics
ENSTA Bretagne

Supervisor

Doctor Jian WAN
Lecturer in Control Systems Engineering
University of Plymouth - AMS

Institution

ENSTA Bretagne
Engineering school
2 rue François Verny, Brest

Academic tutor

Professor Luc JAULIN
Professor in Robotics
ENSTA Bretagne - LabSTICC

September 2020

Acknowledgments

I would like to express my sincere gratitude to my internship supervisor, Doctor Jian Wan, researcher and lecturer in Control Systems Engineering at the University of Plymouth. Despite the quarantine context, my internship has been maintained in France thanks to the sending of all the equipment needed and weekly video conferences. Mr. Wan's availability and advice during this internship allowed me to discover and to further study a large number of key aspects of robotics.

Also, I would like to extend my gratitude to my responsible teacher, Professor Luc Jaulin, researcher at ENSTA Bretagne engineering school, for getting me in touch with Mr. Wan and for facilitating administrative procedures due to the coronavirus pandemic.

I would like to thanks Mr. Benoit Zerr, teacher-researcher in robotics, for guiding me to accelerate administrative procedures.

Also, I want to thank Mr. Fabrice Le Bars, teacher-researcher in robotics, for lending me hardware and authorizing me to keep it for four extra months.

Finally, my thanks also go to my classmate Robin Sanchez, who booked and sent me in emergency extra equipment just before the French lockdown occurred.

Abstract

This report deals with the development of a measurement sailboat for water monitoring, as part of a French second-year engineering school *engineer/research assistant* internship, with the University of Plymouth. The mission consisted of the integration of scientific sensors and the smart measurement path planning of an existing autonomous sailboat.

In this way, the following document is composed of two parts, reflecting the two main issues studied over four months. A first low-level approach allows covering the adding of measurement sensors, the filtering, and the transmission of the acquired data to the sailboat for further analysis. Then, a high-level study proposes various strategies and models to make the sailboat plan a measuring journey in coastal environments, in an autonomous or even smart way.

Résumé

Ce rapport de stage présente le développement d'un voilier de mesure pour la surveillance de l'eau, dans le cadre d'un stage de deuxième année d'école d'ingénieur dit *assistant ingénieur/chercheur*, à l'Université de Plymouth. La mission consistait en l'intégration de capteurs scientifiques ainsi qu'en la planification intelligente du chemin de mesure d'un voilier autonome existant.

Ainsi, le document suivant est composé de deux parties, reflétant les deux principales problématiques abordées au cours de ces quatre mois. Une première approche bas niveau couvre les étapes liées à l'ajout de capteurs de mesure, au filtrage et à la transmission au voilier des données acquises, pour des analyses plus approfondies. Enfin, une étude plus haut niveau propose différents modèles et stratégies pour aider le voilier à planifier un trajet de mesures en milieu côtier, de manière autonome voire intelligente.

Contents

Acknowledgments	1
Abstract	2
Résumé	2
Introduction	4
I Integration of the measurement sensors	5
I.1 Definition of the objectives and constraints	5
I.2 Connection and data collection of the sensors	5
I.3 Transfer of the measurements to the boat through Bluetooth	7
I.4 Data filtering and management for a user interface	9
II Measurement path planning	11
II.1 Definition of the objectives and constraints	11
II.2 Modeling of the autonomous sailboat and its environment	11
II.3 Strategies for an optimal measurement path	14
II.4 Obstacle avoidance approaches in dangerous environments	17
II.5 Reinforcement Learning for a more intelligent planning	18
Conclusion	25
Bibliography	26
Appendices	27
A Complete BLE Attribute Protocol (GAAT) table of the Arduino Bluno	27
B Complete Arduino class diagram	28
C Kalman filter: theoretical and experimental results	29
D Modeling equations and parameters	30
E Effects of wind on path planning	30
F Comparison of Q-Learning reward matrices	32
G Assessment report	35

Introduction

Firstly used exclusively for post-WWII mine-sweeping applications, Unmanned Surface Vehicles (USV) now present a new interest over numerous modern applications and research areas. These vessels without crew are particularly valuable in oceanography. Once equipped with science sensors and navigational instruments, they become more proficient than moored or drifting weather buoys, far cheaper than research vessels or weather ships, and more flexible than commercial-ship alternatives. Among them, sailing ships offer the advantages of being wind-driven and asking little energy to control them, which makes them reliable for long and frequent missions in the ocean and therefore a trusted means for scientific purpose.

Controlling such boats can be remotely operated, partially autonomous, or fully autonomous. The latter group, namely Autonomous Surface Vessels (ASV), has become a common field of research in modern robotics, promising more optimal and even more energy-friendly water monitoring of oceans, with new navigation-related challenges. This is, among other marine systems engineering projects, one central theme studied by the Autonomous Marine Systems (ASM) research group of the University of Plymouth.

In this approach, this report aims to reflect the work accomplished over 4 months, as part of a research assistant internship carried out teleworking with the University of Plymouth. More precisely, in the continuity of researches done by the ASM research group, the focus is based upon *smart monitoring and data analytics of coastal and marine environments using autonomous sailboats*. This theme leads to considering two main objectives: setting up and managing measurements from various sensors on the boat, and proposing a navigation strategy to guide the sailboat to the measurement points.

For safety reasons and due to the coronavirus pandemic, Mr. Jian Wan, my tutor, has ensured the continuity of this internship in France by mailing all academic resources needed and communicating by videoconference. All described codes are available on GitHub [1].

I Integration of the measurement sensors

I.1 Definition of the objectives and constraints

The autonomous sailboat, developed by students and searchers at the University of Plymouth, is not equipped to perform measurements to analyze the Plymouth bay's water quality. This is why a series of specific sensors must be added to the existing architecture of the boat.

Requirements :

The boat must be able to measure and record pH and EC measurements in real-time. All sensors will be contained in a waterproof housing, managed with an Arduino Bluno that will communicate the results to the sailboat using Bluetooth. On the boat, here replaced by a Raspberry Pi board, collected values will be managed and turned into standardized messages with the ROS (Robot Operating System) middleware library, so to be ready to be used for further analysis.

These needs will be covered and described by the next three parts :

1. The various provided sensors must be connected to the Arduino Bluno board, and provide realistic values of the environment. Their measures will have to be saved and be ready to be sent.
2. A communication strategy has to be established from the Arduino Bluno board towards the Raspberry Pi board (sailboat), using their integrated Bluetooth Low Energy (BLE) modules.
3. The data received by the Raspberry Pi board will be processed and turned into standardized messages using the ROS middleware library.

I.2 Connection and data collection of the sensors

Assembly of the hardware components

All the hardware used in this section comes from DFRobot's KnowFlow Basic Kit for Water Monitoring [2]. The kit is provided with Open-source code examples for each sensor, and an assembly guide.

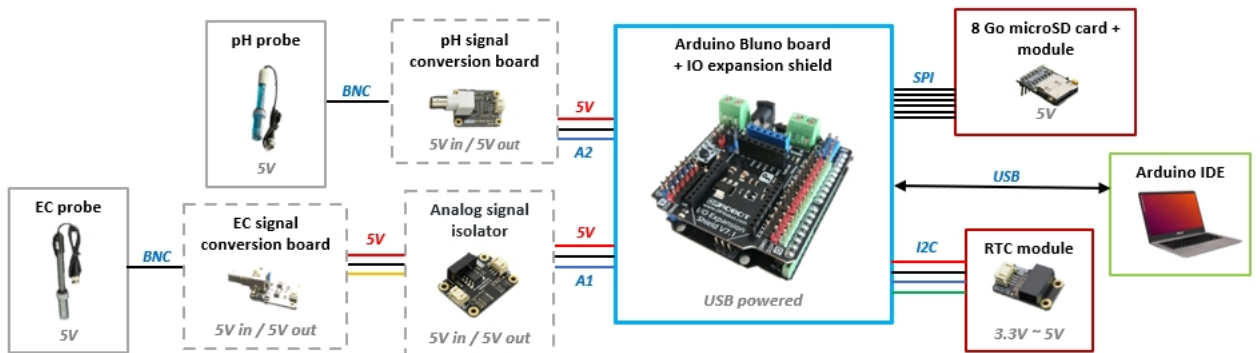


Figure 1: The final mounting layout

The above kit contains:

- 1 Arduino Bluno, identical to an Arduino Uno with an additional BLE integrated module
- 1 IO Expansion Shield (v7.1) to facilitate the connections between components
- 1 Real-Time Clock circuit board (RTC) and its I2C 4-Pin sensor cable
- 1 MicroSD module and 1 8Go MicroSD card
- 1 Analog signal isolator and its analog cable
- 1 pH probe and its circuit board, with buffer solutions
- 1 EC probe and its circuit board, with buffer solutions
- 2 Analog sensor cables

Code organization

To manage the different data coming to the Arduino board, a central script *WaterMonitor.ino* was edited from the Arduino IDE software with its proper local libraries. Each sensor or module is described apart by its class written in the C++ language. More specifically, all measurement sensors are derived classes of the general *ISensor* class with simple shared methods. Another class, *GravitySensorHub*, allows global management for existing sensors and is designed so that future other sensors can be easily added (temperature, dissolved oxygen, and red-ox potential sensors proposed by the same manufacturer).

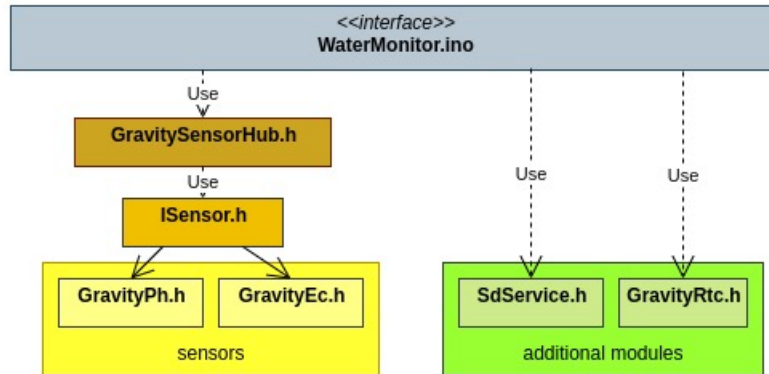


Figure 2: General class diagram on Arduino

Parameters setting of the components

Before any interpretation of the raw data coming to the Arduino, the final data must be categorized. Since precise date and time are often critical for measurements, an option has been added to synchronize the Real-Time Clock (RTC) module with the internal time of the user's computer. Otherwise, the date and hour can be manually set up for convenience. The 2-decimal places-accurate measurements and the time are automatically stored on the 8 Go SD card, written twice at each update to prevent any loss.

Interpretation and adjustment of raw data of pH and EC sensors are performed by the open-source scripts provided by the manufacturer, in which a mean filter is applied to smooth punctual aberrant values. However, to use the measuring instruments at their best performance, a general calibration procedure was taken into account to avoid any malfunction. According to the manufacturer, the probes can reasonably be calibrated every month. When the

Arduino Bluno board is powered and the Serial Monitor opened, the measurements by default take into account previous calibrated values (from the Arduino's EEPROM long-term memory). Calibration can be made at any time, one probe at a time, as long as the code is running on the Arduino board and the Serial Monitor is opened.

Reporting of results

After collection, calculation, filtering, and recording, the resulting values of the pH and EC sensors can be updated approximately every second. This time appears to be reasonable for two sensors but could reach more than 2 seconds with all future sensors added. The current code's global variables use 76% (1558/2048 bytes) of the dynamic memory of the Arduino, which is already a lot but can be explained by the little size of the dynamic memory (SRAM) offered by the Arduino Bluno model (similar to the Arduino Uno model). With additional sensors, the code could be optimized in this direction, or another model among BLE-integrated Arduino boards could be considered (eg. the Bluno Mega 2560 with four times more SRAM).

I.3 Transfer of the measurements to the boat through Bluetooth

Now that the data is correctly arranged, the measurements and their respective date can be sent from the Arduino Bluno and be received by the Raspberry Pi 3 B+ (representing the boat), through Bluetooth Low Energy. In this section, the Arduino board is managed and monitored with the Arduino IDE software through a USB cable, the access to the Raspberry Pi board is performed through ssh protocol from a computer.

How to communicate with BLE-integrated devices

Compared to classic Bluetooth, Bluetooth Low Energy (BLE) is designed to reduced considerably power consumption and cost for a similar communication range. Both Arduino Bluno and Raspberry Pi 3B+ boards have an integrated BLE module 4.0 and appear to be compatible.

Any communication between two devices needs to be initiated in two stages:

- Device discovery is done through the Generic Access Profile Protocol (GAP), depending on if the devices have either a peripheral or central role. A peripheral (Arduino, sensors, or wearable...) advertises itself and waits for a central to connect to it, whereas a central (Raspberry Pi or computer) scans other devices. Once connected, the peripheral becomes a slave and the central a master.
- After discovery, device-to-device communication occurs through the Generic Attribute Protocol (GATT). It establishes common operations and a framework for the data between the client (Raspberry Pi) who can READ/WRITE to the server, and the server (Arduino Bluno) who can NOTIFY/RESPOND to the master and who possesses the information to share. The Attribute Protocol (AT) table specifies all resources (services and characteristics) proposed by the server.

Finally, based on the Attribute Protocol provided by the Arduino Bluno, the transmission becomes:

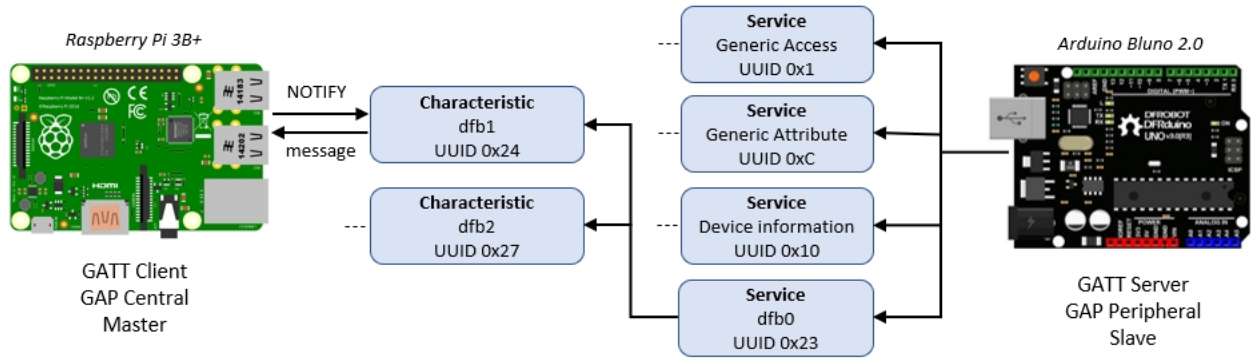


Figure 3: Simplified BLE protocol used between the two devices

Data transmission and receiving

My tutor, Mr. Jian Wan, advised using the Bluez Linux tool and Bluepy Python library as they have proved to give good results for one of its Ph.D. students.

As a first step, the Bluez software installed on the Raspberry Pi allows introducing the two devices, by making the Arduino Bluno trustworthy. It also provides the MAC address of both, which is equivalent to a unique personal identifier.

The Arduino Bluno automatically transmits through BLE the data printed on its Serial Monitor. The device is set on the USB-UART BLE transparent mode to allow debugging, the minimum and maximum connection intervals are set up to 10 ms for better compatibility with a computer, and the baud rate is fixed at 115200 as recommended by the manufacturer. However, the transmissible data is limited in size, so a special format is employed to restrict each message to its minimum so it is sent entirely. Three different messages are sent every 300 milliseconds at each update (lasting 1 second):

- “M–d–y–h–m–s–” : Date and time
- “pH–tp–EC–” : pH, temperature and electronic conductivity
- ”Do–Or–” : dissolved oxygen and oxidation-reduction potential

The above dashes represent the effective values. Their accuracy is not limited in the messages, but the manufacturer decided to set it up to two decimal places. The last message is not useful at this point, but it anticipates the additions of the remaining sensors.

On the Raspberry Pi 3B+, a *ble_interface_node.py* Python script using the Bluepy library manages the pairing, the scanning, and reception of notifications sent by the Arduino. The data is split and updated into variables in this same script.

Result of transmission

Using BLE, communication between the devices is feasible over at least 20 meters without data loss (70 meters theoretically, but not tested because of material reasons). On the receiver’s side, very rarely, the three different messages may not always arrive every second (as normally sent from the Arduino), but it does not compromise the global data reception (one occurrence every minute at most). If the data is not directly received, the last saved values are used. The BLE is therefore an appropriate transmission protocol for the goals of this internship.

Finally, one crucial aspect is blocked by the BLE compatibility between the Raspberry Pi and the Arduino Bluno. It seems indeed that a specific USB dongle proposed by DFRobot allows getting rid of manually maintaining the Serial Monitor open, to allow data sending from the Arduino. This issue could be studied to reach a user-free method for BLE communication.

I.4 Data filtering and management for a user interface

Finally, once collected by the Raspberry Pi, the data must be correctly sorted and filtered before being sent to the user. A middleware architecture will be used to showcase general steps and to better identify data flows.

Middleware architecture

In this section, the middleware used is ROS melodic v. 1.14.5, installed on a Raspberry Pi 3B+ running on Ubuntu 18.04 distribution. The use of middleware is needed to create three interfaces: one for receiving and sorting the data previously acquired by BLE transmission, one for data filtering to prevent any jumps in values due to many existing sources of errors, and a final one that receives the filtered data and transmits it to the user.

Then, using the Rviz tool, the following node diagram is obtained :

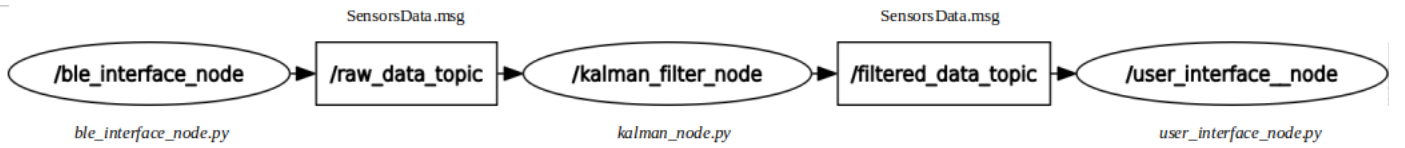


Figure 4: ROS node diagram on the Raspberry Pi

The two services `/raw_data_topic` and `/filtered_data_topic` ensure the flow of information. Each carries a custom message of type `SensorsData` to send all information at once.

Structure of `SensorsData.msg` :

Type	Name
float64	ph
float64	conductivity
float64	temperature
float64	dissolved_oxygen
float64	redox_potential
string	date
bool	is_connected

Kalman filter node

The raw data received via BLE may present some inconsistencies, either due to measurement errors from the sensors, or data losses from BLE transmission, or problems of update synchronization. A Kalman filter, as wanted by Mr. Jian Wan, could reduce these errors by correcting and predicting behaviors of the values.

Let x be the array of measurements to analyze. Then, the system chosen to reflect the measurements process, at step k , is :

$$\begin{cases} \text{model :} & x_{k+1} = A \times x_k \\ \text{measurements :} & y_k = C \times x_k \end{cases} \quad \begin{cases} \text{estimated model :} & x_{es,k} = A \times x_k + a \\ \text{estimated measurements :} & y_{es,k} = C \times x_{es,k} + b \end{cases} \quad (1)$$

with:

$$\begin{cases} x = & \begin{bmatrix} pH_{value} & EC_{value} \end{bmatrix}^T \\ A = & I_2 \\ G_x = & 0.2 \times I_2 \end{cases} \quad \begin{cases} C = & I_2 \\ G_a = & 0.2 \times I_2 \\ G_b = & 0.5 \times I_2 \end{cases} \quad (2)$$

where x is the state vector, G_x the estimated error on the initial state, G_a the estimated error on the model, G_b the estimated error on each measurement, and I_2 the identity matrix.

For more realistic equation models, some noises were imagined :

- A centered Gaussian white noise a on the model, to estimate the error on the values made by the sensors. The resulting covariance matrix of the process noise is G_a , with a variance of 0.5 (reflects a medium uncertainty on the model).
- A centered Gaussian white noise b on the measurements, to estimate the error on the values made during the BLE transmission. The resulting covariance matrix of the observation noise is G_b with a variance of 0.2 (reflects a light uncertainty on the measurements).

Consequently, the filter used in `kalman_filter_node` is:

Algorithm 1 Kalman Filter

Input: $x, y, A, C, G_x, G_a, G_b$

Output: $x_{pred}, G_{x,pred}$

Correction

- 1: $S \leftarrow C \cdot G_x \cdot C^T + G_b$
- 2: $K \leftarrow G_x \cdot C^T \cdot S^{-1}$
- 3: $\tilde{y} \leftarrow y - C \cdot x$
- 4: $G_{up} \leftarrow (I_n - K \cdot C) \cdot G_x$
- 5: $x_{up} \leftarrow x + K \cdot \tilde{y}$

Prediction

- 6: $x_{pred} \leftarrow A \cdot x_{up}$
 - 7: $G_{x,pred} \leftarrow A \cdot G_{up} \cdot A^T + G_a$
 - 8: **return** $x_{pred}, G_{x,pred}$
-

Results of data processing

When launching the BLE transmission and the ROS process, data manipulation appears to be easier. Intermediary nodes allow tests and debugging, but the final user is not affected by them since its interface only receives the data after filtering. Also, filtering does not delay the data flow, the user interface is updated every second.

A graphical tool has been implemented to better acknowledge the impact of the Kalman filter in its corresponding node. After theoretical and practical tests, the filter does not always predict exactly the “true” sensors values, but still is nearer to the real ones than “raw” values are. Its principal effect remains to smooth pH and EC measurements, but it cannot be exploited further because of the impossibility of predicting pH and EC evolutions over time (they are therefore designed as constants in the model). In future works, the Kalman filter should be updated to process additional sensors.

II Measurement path planning

II.1 Definition of the objectives and constraints

Now that the sailboat is equipped to perform and communicate measurements on water quality, it still needs to learn where and how to take the measurements so that water monitoring could be automated on an entire area.

Requirements:

Given a set of points to visit in a known environment, the sailboat must be able to plan a safe path to take measurements. The planning strategy will take into account the direction of the wind and existing obstacles (rocks, coats) and has to be calculated under a reasonable amount of time. The solution must be optimized so that the path is as short, fast, and energy-efficient as possible.

These needs will be covered and described by the four following parts :

1. The main actors in the problem must be modeled. The boat model and simulation will reflect the dynamics of a simplified sailboat, its controller will be implemented in ways to steer this model with way-points guidance. The environment should also be represented, and reflect constraints brought by the wind (in the first instance).
2. The optimization problem has to be identified to propose several simple solving strategies.
3. For a more realistic resolution, constraints linked to borders and obstacles will be taken into account to start to implement avoidance methods.
4. Finally, for better results, long-term strategies using Artificial Intelligence could be considered.

II.2 Modeling of the autonomous sailboat and its environment

To facilitate path planning, the problem modeling should take into account all constraints associated with the sailboat’s properties, the wind’s speed and direction as well as the measurement area’s borders and obstacles.

Sailboat modeling

The sailboat model used comes from John Melin’s *Control and state-estimation for an autonomous sailboat* [3]. It assumes that the boat has a unique effective sail and evolves in a 2D-space, where the influence of waves and current is neglected and its velocity is assumed to

be small.

Among all parts of a standard sailboat, the paper notably relies on three main components: the hull whose characteristics serve for the boat's state vector, the angles of the sail and the rudder for the dynamics and control.

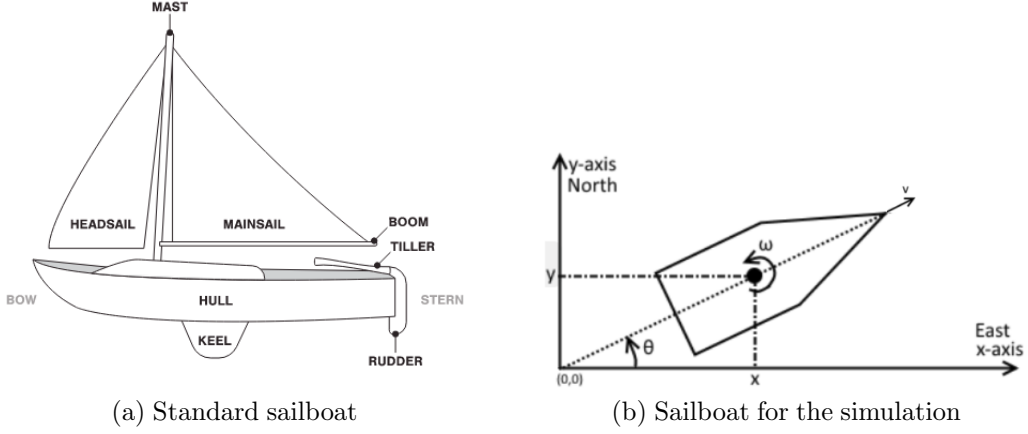


Figure 5: Sailboat transformation for modeling

Sailboat's variables

- The state vector of the boat, based on the hull's properties, is defined as : $X = [x, y, \theta, v, w]^T$ with the sailboat's position (x, y) , velocity v , heading θ , and rotational speed w given in a North-East-Up reference frame.
- δ_r is the angle of the rudder.
- δ_s is the angle of the sail which is proportional to the length of the mainsheet.
- Other variables (p_0, \dots, p_{10}) are intrinsically linked to the boat's shape and dimensions and remain constant over time. They have been changed to fit the Plymouth student's sailboat's dimensions.

Wind's variable

The wind is depicted accordingly to two frames for convenience :

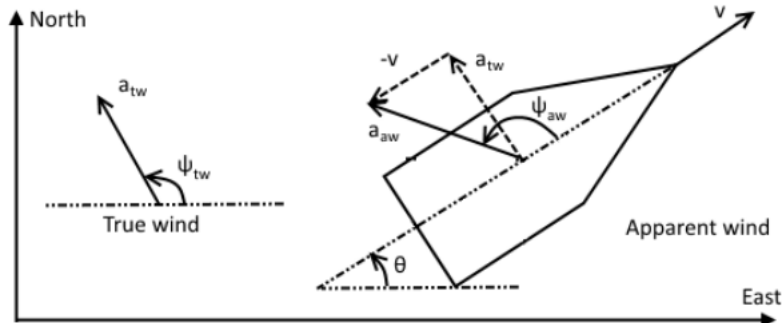


Figure 6: Modeling of wind's parameters

- True wind: W_{tw} , the coordinates in the global frame of the wind of speed a and direction ψ .
- Apparent wind: W_{aw} , the (cartesian or polar) coordinates of the wind perceived by the boat in its frame (relative to its own direction).

Evolution equations

Considering the sail force, the rudder force, and the friction, the hull, the rudder and the sail can evolve dynamically for the wind's constraints (consult the appendix for more details).

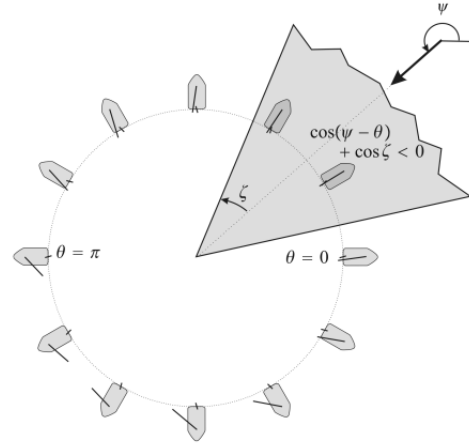
Control of the sailboat

The following description refers to the strategy proposed by Mr. Jaulin and Mr. Le Bars in *A simple controller for line following of sailboats* [4], in which they introduce a pragmatic approach influenced by a potential field strategy for the sailboat model previously described.

With this method, the boat is progressively attracted by the line to follow until it fits it. But some directions, like paths facing the wind, are not feasible for the sailboat. These unfeasible courses form a no-go zone (painted in grey), that can be avoided thanks to a keep close-hauled strategy. Therefore, instead of directly following a line, the boat will adopt a zigzag trajectory to target the goal while staying at the frontier of the allowed zone.

Function in: $\mathbf{m}, \theta, \psi, \mathbf{a}, \mathbf{b}$; out: $\delta_r, \delta_s^{\max}$; inout: q
1 $e = \det \left(\frac{\mathbf{b}-\mathbf{a}}{\ \mathbf{b}-\mathbf{a}\ }, \mathbf{m}-\mathbf{a} \right)$
2 if $ e > \frac{r}{2}$ then $q = \text{sign}(e)$
3 $\varphi = \text{atan2}(\mathbf{b}-\mathbf{a})$
4 $\theta^* = \varphi - \frac{2 \cdot \gamma_x}{\pi} \cdot \text{atan} \left(\frac{e}{r} \right)$
5 if $\cos(\psi - \theta^*) + \cos \zeta < 0$
6 or ($ e < r$ and $(\cos(\psi - \varphi) + \cos \zeta < 0)$)
7 then $\bar{\theta} = \pi + \psi - q \cdot \zeta$.
8 else $\bar{\theta} = \theta^*$
9 end
10 if $\cos(\theta - \bar{\theta}) \geq 0$ then $\delta_r = \delta_r^{\max} \cdot \sin(\theta - \bar{\theta})$
11 else $\delta_r = \delta_r^{\max} \cdot \text{sign}(\sin(\theta - \bar{\theta}))$
12 $\delta_s^{\max} = \frac{\pi}{2} \cdot \left(\frac{\cos(\psi - \bar{\theta}) + 1}{2} \right)$.

(a) Controller algorithm



(b) Dead zone range to avoid

Figure 7: The controlling strategy of Mr. Jaulin and Mr. Le Bars

Finally, the algorithm groups the two strategies: a potential field strategy if the boat is guaranteed to stay in the allowed zone, and a closed-hauled strategy if it risks to end up in the no-go zone. For more details on each step, consult the paper.

Environment modeling

Visiting a set of measurement points refers to an already well-known optimization theme commonly illustrated by graph theory: the Traveling Salesman Problem (TSP), which naturally leads to using graph theory to represent the environment to explore. Each measurement area, visited by the boat, is therefore composed of:

- vertices, with unique labels, for the different points to visit
- edges, for the existing paths that link each point to other points
- weights, as efforts for traveling from one point to another (generally the distance). The smaller the weight, the easier the travel.

A simple TSP generally asks for a complete undirected graph, in which every pair of distinct vertices is connected by a unique edge with a constant weight (all directions are possible and the traveling from one place to another costs the same). Considering a sailboat this time, the orientation of the wind prevents the boat from moving in all directions.

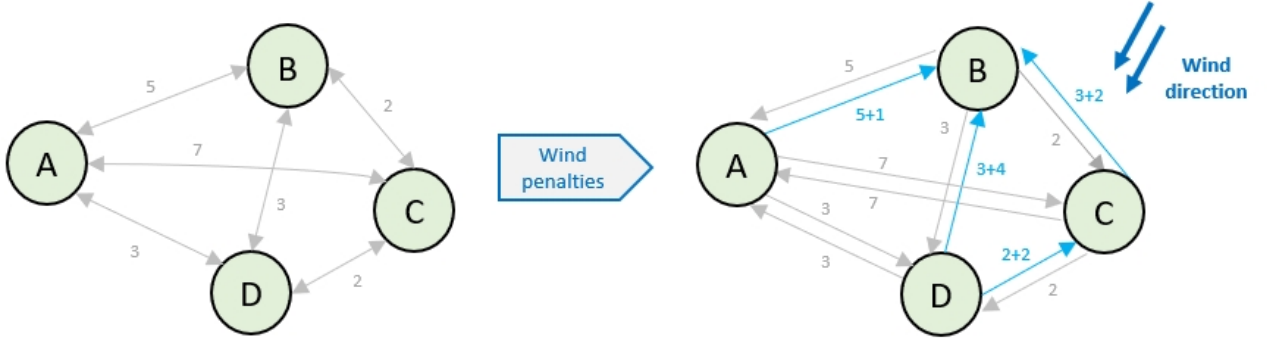


Figure 8: Effect of wind penalties on a graph

Consequently, the graph of the problem becomes asymmetrical (oriented). Moreover, it is assumed that all measurement points are still reachable from every other point, but that the effort between two points differs depending on the direction (a wind penalty increases the distance). Then, a complete digraph is obtained to model the problem.

Models implementation

In practice, three Python classes group the previous modeling:

- Boat class: the boat model with its controller. The simulation of the boat is updated with a Runge-Kutta fourth-order method for trajectory evaluation.
- Graph class: graph objects with possible solving strategies. All the corresponding data is stored in lists instead of matrices, to reduce the complexity of the calculations when the number of points to consider increases.
- Area class: to easily construct a set of points. The user can either choose between automatically built grid and circle shapes with homogeneously distributed points, or manually create an environment.

II.3 Strategies for an optimal measurement path

Now that the sailboat and the measurement zone are defined and modeled, various algorithms can be implemented to solve this specific Traveling Sailboat Problem. Because it becomes very difficult to use Dijkstra or Floyd-Warshall algorithms in this case, and it is not possible to have recourse to the Christofides algorithm (triangle inequality not guaranteed), alternative strategies should be built.

Edges building

Accordingly to what has just been introduced, edges can be easily created by linking all possible pairs of points to visit. This method offers lots of combinations, therefore a better chance to find the optimal solution. However, in the case of iterative solving algorithms, more effort can be needed to find a good path. So to connect the vertices with their closest neighbors, a new method based upon Delaunay triangulation is proposed so that long edges are automatically disqualified.

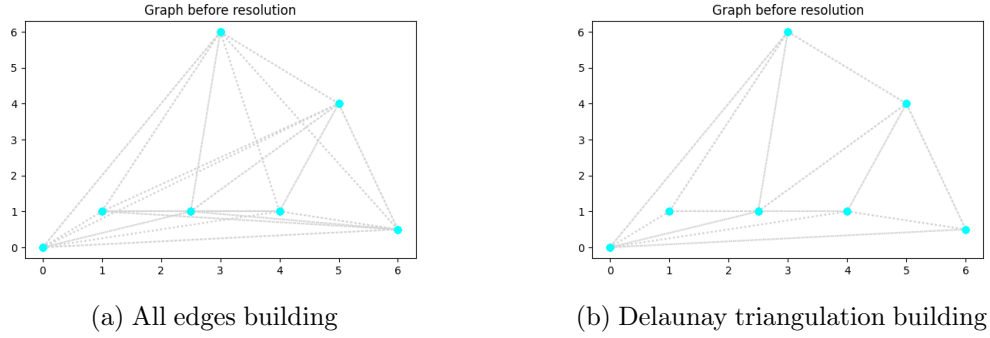


Figure 9: Two edges building methods applied to the same graph

From the obtained graphs, it is easy to see that the Delaunay triangulation already facilitates the future work of solving algorithms. However, its major weakness is the possible absence of a final edge for returning to the starting point: an alternative edge (linking the ending point with the starting one) is created in this case, but it may encounter an obstacle.

Solving algorithms

Four algorithms of various complexity are taken on. Their efficiency is evaluated from the distance of the final path proposed.

- Random strategy: a basic algorithm that compares the scores of two paths constructed from swapping two randomly-sectioned points.
- Loop strategy: same as random strategy, but all pairs of swapping points are evaluated.
- Nearest-Neighbor strategy: a one iteration greedy algorithm looking for the best weight at each step.
- Genetic strategy: a random-based algorithm that brings in mutations (errors not respecting short-term rewarding but allowing long-term rewarding).

The efficiency of solving algorithms

The algorithms have been tested on various shapes, for various wind directions, and edges all connected.

For several shapes and wind directions, the Nearest-Neighbor strategy seems to provide the optimal existing path. A trend for straight and parallel sections of path also emerges with other algorithms, but they cannot compete here with the score found by the first one, certainly because the process got stuck at a time and it asks too many changes to significantly decrease the total length. The loop strategy converges faster than the random strategy. The genetic strategy needs more time than the others, but already offers a better score than the random

and loop algorithms.

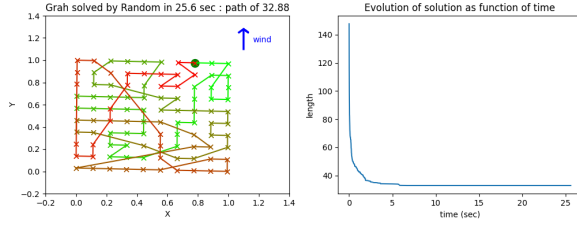


Figure 10: Random strategy

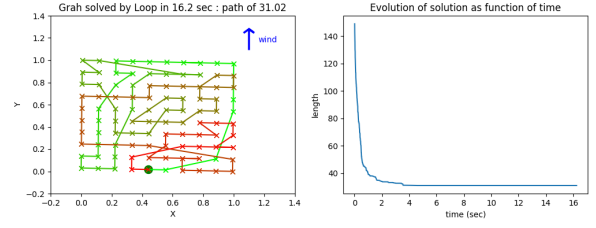


Figure 11: Loop strategy

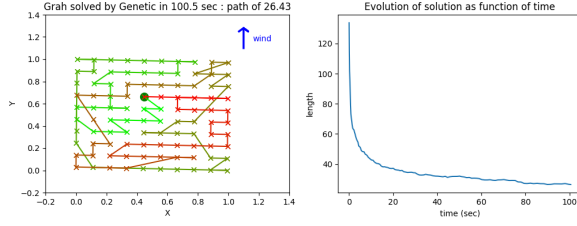


Figure 12: Genetic strategy

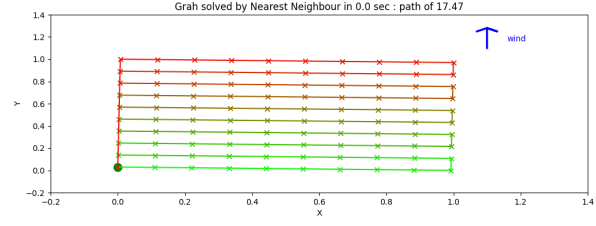


Figure 13: Nearest-Neighbor strategy

Wind penalties applied on the edges, according to their vertices orientation and the wind direction, clearly affect the solutions proposed by each solving strategy. The same trends are observed for different wind directions, with the two edges building methods.

Depending on the environment size and the parameters chosen for the algorithms, the random, loop and genetic algorithms take between one and four minutes to propose a decent solution. On the contrary, the Nearest-Neighbor strategy proposes a path immediately.

From these observations, the Nearest-Neighbor algorithm is undoubtedly the algorithm providing paths with shortest lengths for various scenarios, in record time.

The efficiency of edges building methods

From this step, the Nearest-Neighbor strategy is selected as the most efficient solving algorithm. The two building methods are tested on the two shapes for different wind directions.

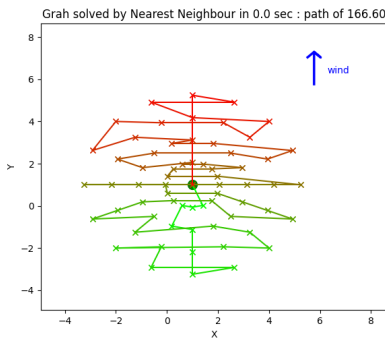


Figure 14: Automatic edges building

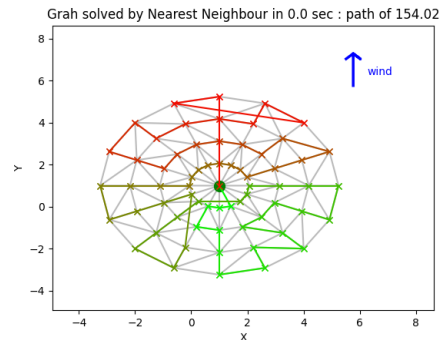


Figure 15: Delaunay triangulation building

For grid-shaped areas, the Delaunay method does not provide better results than the automated completion method. This is notably due to the solving algorithm that, for this geo-

metric configuration, directly disqualifies the longest edges.

For circle-shaped areas, however, the Delaunay strategy significantly improves the score of the solution. Because if edges are all connected, the geometric configuration tricks the solving algorithm at some points. So, the Delaunay method is insightful for such complex geometry, and may even propose paths with fewer difficult U-turns.

II.4 Obstacle avoidance approaches in dangerous environments

Now that the most relevant edges building method and solving strategy have been selected, the sailboat must now be prepared for more dangerous exploration allowing measurements near coasts borders, and the presence of irregular obstacles.

Coastal shape

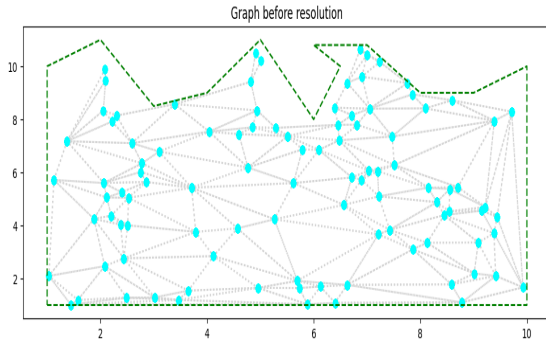


Figure 16: Graph before resolution

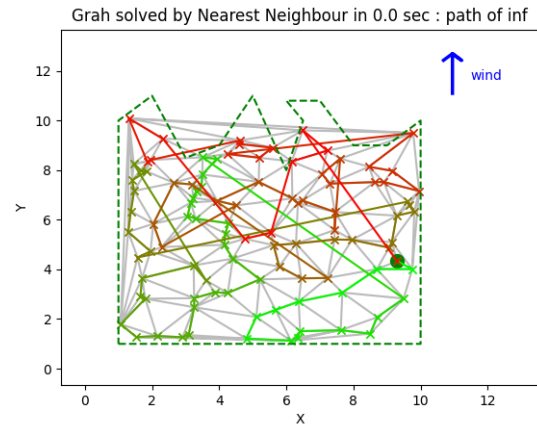


Figure 17: Graph solved

Another more realistic coastal-shaped area is available in the Area Python class. Now, when building edges, the border of the measurement area will be taken into account to delete all edges passing across risky zones. If, because of the logic of algorithms, the path effectively encounters a frontier, a detour will be planned like for other normal obstacles.

Obstacle avoidance strategy

Short detours are a solution to circumvent small obstacles without lengthening the path too much. These deviations are calculated once the path is planned by the boat, so it allows keeping a maximum number of edges without interfering directly with solving algorithms.

A safety distance is fixed to bypass the obstacles within a certain minimum perimeter, as explained and justified in *Optimization-Based Motion Planning With Obstacles And Priorities* paper of Greiff and Robertsson [5]. With this constraint, the shortest possible detour replaces the problematical portion of the initial path.

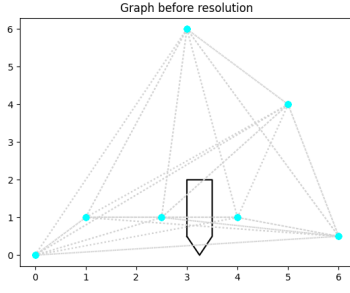


Figure 18: Initial graph

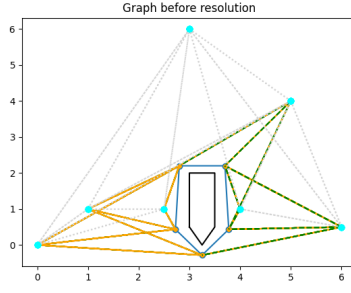


Figure 19: Avoidance

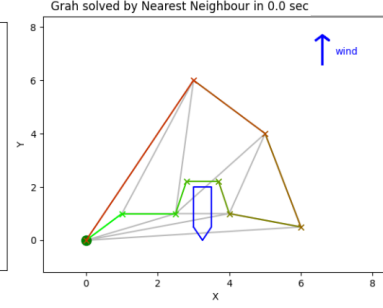


Figure 20: Resolution

This strategy offers good results on simple graphs, as well as for graphs with hundreds of points. However, if bigger obstacles show up in the next environments, maybe modifying directly the graph and/or the solving algorithms could avoid useless long detours.

Voronoi diagram simple avoidance

The previous strategy is ideal in environments where measurements are necessary. In the case of simply crossing dangerous areas, a common approach inspired by the Voronoi diagram permits to quickly draw safe paths across multiple obstacles, by generating a graph of points. After adding wind penalties on some edges of the obtained graph, the Dijkstra algorithm then determines the quickest route between any pair of points.

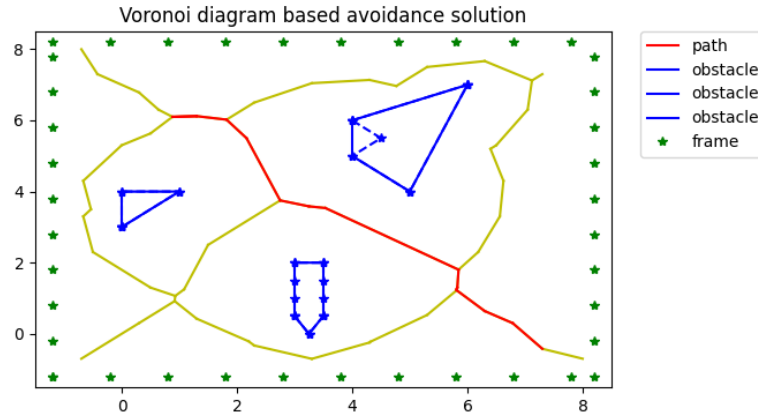


Figure 21: Voronoi diagram-based avoidance strategy

II.5 Reinforcement Learning for a more intelligent planning

All paths calculated with previous algorithms do not necessarily offer the most optimal solutions. Because, either time of calculation is limited, which bothers random-based strategies, or the main approach mostly relies on a series of instant reward decisions (Nearest-Neighbor strategy). Then, further approaches such as Machine Learning could provide more interesting long-term strategies for the sailboat.

Which Machine Learning for the sailboat?

Main Machine Learning algorithms can be divided into three major groups, depending on the task asked:

- Supervised Learning (task-driven) for classification into groups and prediction of future values,
- Unsupervised Learning (data-driven) for indirect identification of clusters or patterns,
- Reinforcement Learning (learn from mistakes and reward) to understand how to act in a given environment.

In path planning problems, evaluating a route with a score makes sense (length or obstacle collisions), the optimal path is not always known, and finding general rules for performing a good path, with directions alone at disposal, in an unknown environment, seems complicated to be implemented. These reasons naturally lead to the use of Reinforcement Learning.

Neural networks turn out to require experience to be comprehended, and beyond this, no model approximately meeting the conditions set by the sailboat has been found. For convenience, methods using Q-Learning will be developed in this section.

Q-Learning philosophy

Reinforcement learning involves an agent s , a set of states S , and a set A of actions per state. By performing an action $a \in A$, the agent transitions from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score). Q refers to the function that returns the reward used to provide the reinforcement and can be said to stand for the quality of an action taken in a given state.

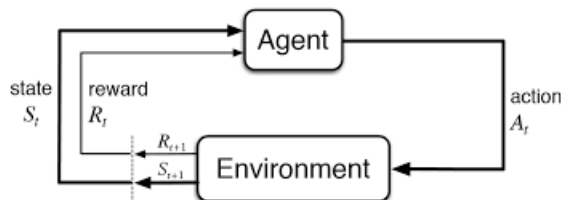


Figure 22: Q-Learning principle

For a finite Markov decision process (a known number of possible positions), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over all successive steps, starting from the current state.

When Q-learning is performed, a Q-table is created. This matrix follows the shape of [state, action] with arbitrarily-initialized values. Then, these Q-values are updated after each episode and stored. This Q-table becomes a reference table for the agent, to select the best action based on the Q-values. The process is implicitly configured with:

- ϵ the random explorer (either to avoid loops and encourage exploration or to encourage exploitation of correct behaviors)
- e the horizon (number of total training)

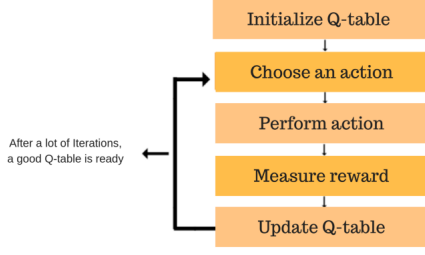


Figure 23: Q-Learning implementation

State	Right	Left	Up	Down
1	0	0.31	0.12	0.87
2	0.98	-0.12	0.01	0.14
3	1	0.10	0.12	0.31
4	0.19	0.14	0.87	-0.12

Figure 24: Example of a Q-Table

At each step t , the Q-Table is updated according to this Bellman's equation:

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}_{\text{new value (temporal difference target)}}$$

temporal difference

where:

- α is the learning rate (how important are the new rewards compared to the previous ones)
- γ is the discount factor (to balance immediate and future rewards)

The better these hyperparameters $(\epsilon, e, \alpha, \gamma)$ are calibrated, the more efficient the algorithm.

Reinforcement learning applied to an autonomous sailboat

The main actors shown in the previous section are kept, but they need to be adapted to enable a Q-Learning resolution. Their new modeling is inspired here by *High-Level Path Planning for an Autonomous Sailboat Robot Using Q-Learning* written by A. Junior, D. Henrique dos Santos, A. Fernandes de Negreiros, J. Boas de Souza Silva and L. Gonçalves [6].

First, the Q-Learning agent corresponds exactly to the same sailboat model as depicted before, used for static algorithms. However, only general dynamics will be considered: three degrees of freedom in a 2D plane, with waves, currents, and gravitational forces neglected, and a no-go zone present.

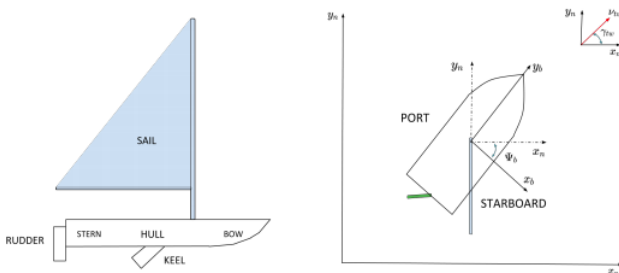


Figure 25: The sailboat agent

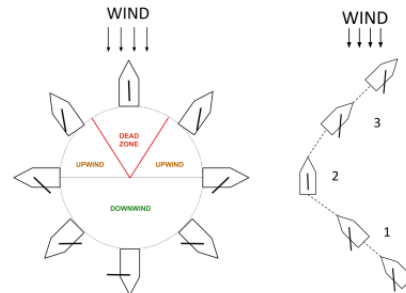


Figure 26: Simplified dynamics

Also, the authors have chosen to represent the zone of measurements as a discretized environment, instead of a graph. This directly allows a simple strategy for avoiding the obstacles and borders (by discretizing them too). Each cell of the new environment is numbered to be associated with a unique state.

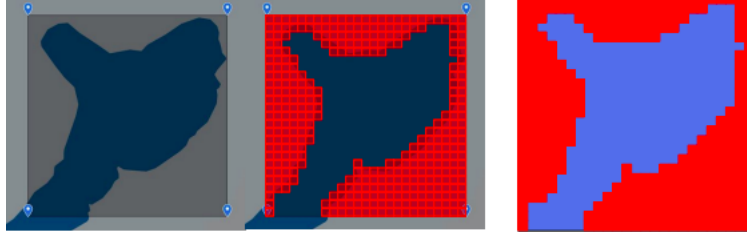


Figure 27: Discretization of the environment

In this way, representing the map as a grid restrains the number of possible actions to 8 directions (horizontal, vertical, or diagonal moves). Some of these actions must be removed to acknowledge the no-go zone due to wind.

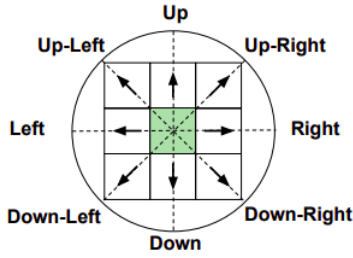


Figure 28: The sailboat agent

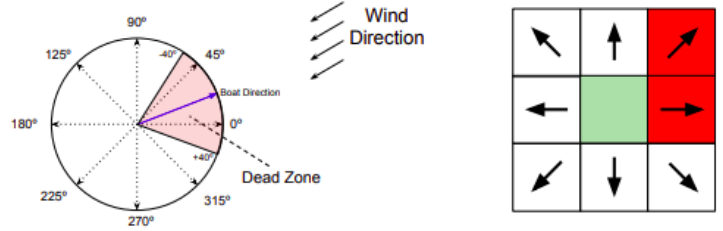


Figure 29: Set of actions

Finally, the authors have imagined a reward matrix based on the inverse distance between the cells of the environment. All distances are calculated for each state from the ending point (figure 30a), then inverted, and added with the maximum found distance to obtain a score (figure 30b). In this way, the ending point has the highest reward of the grid. Cells with obstacles have a score of -1000.

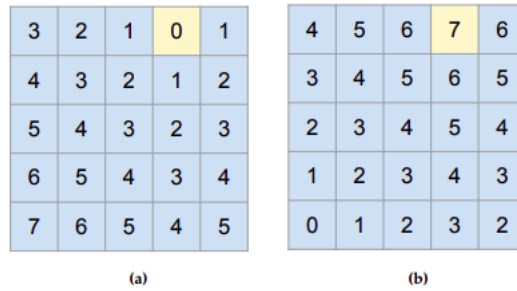


Figure 30: Distance matrix (a) and its associated gradient reward matrix (b)

Surprisingly, instead of simply being a combination of neutral, positive, and negative rewards, the reward matrix used in *High-Level Path Planning for an Autonomous Sailboat Robot*

Using *Q-Learning* relies on a various set of reward values, determined by a climb-gradient-like method. This new rewarding method may bring real underlying advantages. Then, the mission becomes to set up some tests, to understand if this method is more relevant than classic ones for two cities, or if it is more convenient for problems with even more cities.

Improving the reward matrix for two points

Classic reward matrices used in Q-Learning assign up to three different values: a neutral value (-1) for standard cells, a consequent negative score for obstacle cells (-100), and a high positive reward (+100) on the cell to reach.

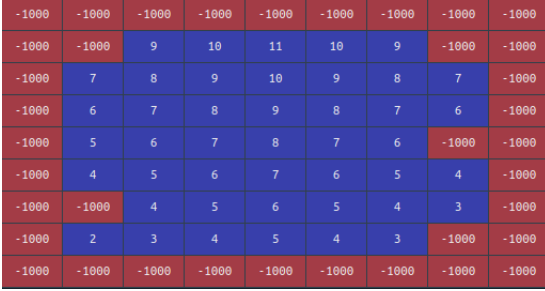


Figure 31: Gradual reward matrix

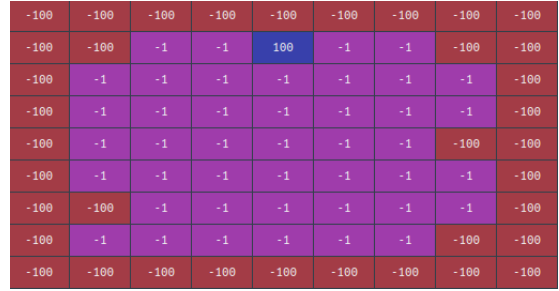


Figure 32: Classic reward matrix

With these configurations, the reward matrix presented in the article (figure 31) seems to encourage an instant reward strategy (discount factor close to 0), whereas the classical one (figure 32) is designed to find the shortest path by penalizing all normal cells and to encourage long-term strategies thanks to its unique positive cell (discount factor close to 1).

Initializing all Q-Table values to zero allows obtaining a solution with the classic reward matrix, whatever the discount factor. However, in order to use the gradient matrix reward, the Q-Table must be set up differently: all the values in the row corresponding to the goal's state must be higher than a certain value (depending on the map, and distance from the starting point). Otherwise, it is not sure that the agent arrives at the objective cell. This aspect may be inconvenient to organize automatic path planning.

The following two results are run with Python and use the following hyperparameters: $\epsilon = 0.3$, episodes = 10000, $\alpha = 0.8$, $\gamma = 0.7$. They are displayed with a graphic tool especially implemented, to spotlight the behaviors of the Q-Table and the general convergence of accumulated errors.

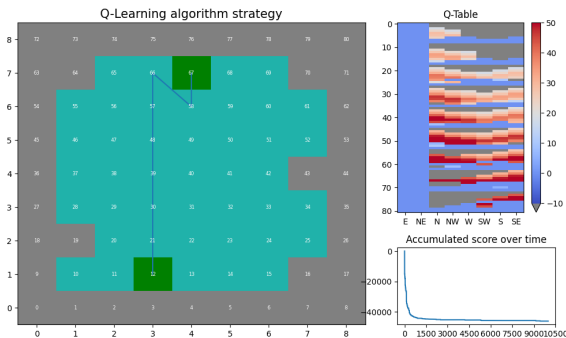


Figure 33: Gradual reward resolution

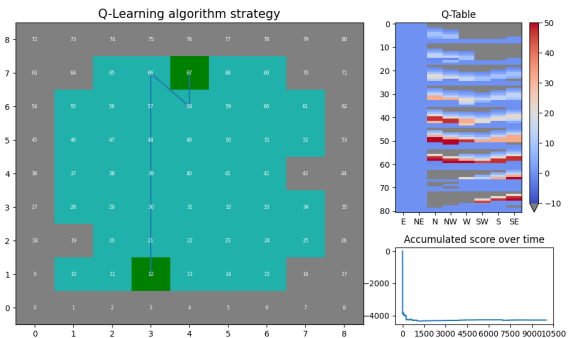


Figure 34: Classic reward resolution

The environment and the paths traveled are shown on left, with cells numbered accordingly to their state. The Q-Table appears on the upper-right corner with states as rows and actions as columns (disabled actions are identifiable by their constant color over time). The accumulated error as a function of episodes is displayed on the lower-left corner.

Using the classic reward matrix (figure 34) tends to make the algorithm converge more quickly. Both of them result in the same optimal solution for this map and configuration. However, with different hyper-parameters, the gradient reward matrix (figure 33) does not always propose a solution (because of infinite loops due to similar values in the Q-Table). By analyzing the final appearance of the Q-Tables, it seems that values created by the gradient rewarding are quite gradual and continuous, while those of the classic rewarding look more heterogeneous. This may reflect a confidence in the path found among other possibilities.

Further comparison between the reward systems for two points

In the article, the authors have organized a series of different scenarios to test the robustness of their model. These scenarios can be reused to compare further, step by step, the two reward systems. The same hyperparameters, as described in the paper for each scenario, will be used. The full comparison of the gradient reward system and the classic reward system are presented in the appendices.

This further comparison finally leads to the same observations as the previous one (Q-Table final appearance, convergence speed). For nearly identical results, the classic reward system could replace the gradient one if convergence speed is a crucial criterion.

Test of the reward systems for three points and more

To return to the initial objective and solve the Traveling Sailboat Problem, the Q-Learning strategy should expand to more measurement points. Taking into consideration more than two points on the map forces to update the two previous reward systems. Ideally, the optimal shortest path should go through every measurement point only once. By calculating the rewards accordingly to each interesting point's position, this condition is not verified: the boat can return to the same point, again and again, to maximize its score while ignoring the other points to discover. One possible solution is to update the reward matrix after having visited a point.

Algorithm 2 Reward Matrix updating

Input: $R0, all_points_to_visit$

Update for one episode

- 1: $R = R0$
 - 2: $points = all_points_to_visit$
 - 3: **while** all points have not been visited **do**
 - 4: **if** a measurement point P is reached **then**
 - 5: $R = R(points - P)$
 - 6: $points = points - P$
 - 7: **end if**
 - 8: **end while**
-

The reward matrix, therefore, becomes dynamic. For initialization, this matrix R_0 is built to consider all points with the same usual score, so to not influence the agent in its path planning. While the Q-Learning algorithm is running, it can be updated with the previous algorithm. This updating method can be applied to both previous reward systems.

From observations, the classic reward system could converge less frequently because of the appearance of random noise with time. This perturbation may come from updating the reward matrix and can confuse the agent. Another source of error occurs with both algorithms because it is impossible to force the order in which each city is discovered. One time, the path will begin by point A and finish on point B, another time it will be the opposite. Considering the wind besides, the algorithms generally tend to find a "blurred path" close to the solution with some points of interest missing in the optimal found path.

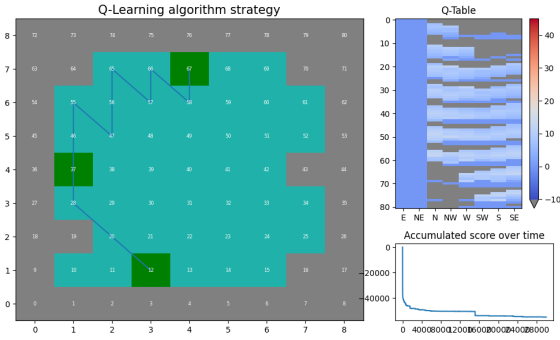


Figure 35: Gradual reward resolution

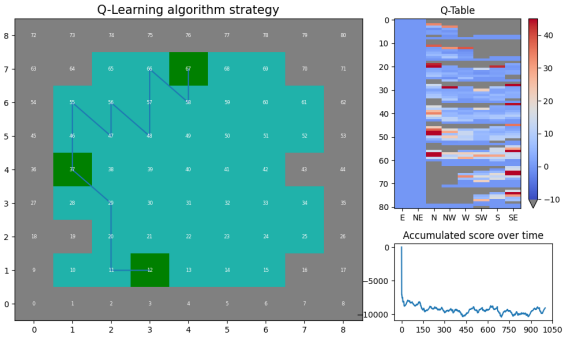


Figure 36: Classic reward resolution

As such, the gradient reward system imagined in the paper seems to be more reliable for this issue. Cases with more points have been tested, but the results are less convincing and take even more time for a single training (more than 5 minutes for 10 000 steps).

Hence limitations of Q-learning emerge when directly influencing the Q-table updating. Although this method is robust for two points problems and offers the advantage of simply avoiding obstacles and borders, it remains very sensible to applications outside its normal use. Working with Deep Q-Learning, a more complex but versatile Machine Learning approach seems unavoidable to deepen the modeling proposed in the studied article.

Conclusion

Although this internship was entirely carried out teleworking, it nonetheless allowed me to study an interesting project in its whole. From hardware constraints to high-level fields of study, the various themes often asked for the application of recently acquired theoretical knowledge at ENSTA Bretagne, while making me discover new fields, which was particularly gratifying.

The various tests conducted without the boat, for the adjustment and the integration of the measurement sensors, could be deepened on the real sailboat by future students. That being said, the sensors provided and data transmission through Bluetooth Low Energy have proved to be robust means which meet the needs of water monitoring.

Also, working on an irresolvable optimization problem such as path planning, revealed to be a significant but stimulating challenge. I was then allowed to study very recent papers, to be confronted with specific current issues faced by modern robotics, and to propose partial solutions to it. It would be really interesting if, future students who will resume the project could address in more depth Machine Learning applied to the path planning for an autonomous sailboat.

I highly recommend this internship to those who would like to strengthen their hardware knowledge, or who want to study an innovative subject such as autonomous sailboats.

Bibliography

- [1] Agathe Archet. Github repository. <https://github.com/AgatheArchet/smart-water-quality-monitoring>.
- [2] DFRobot company. Kit knowflow basic for water monitoring. <https://www.dfrobot.com/product-1649.html>.
- [3] Jon Melin. *Modeling, control and state-estimation for an autonomous sailboat*. PhD thesis, Uppsala Universitet, 2015.
- [4] Fabrice Le Bars Luc Jaulin. Proceedings of the 5th international robotic sailing conference. In Springer Eds, editor, *A simple controller for line following of sailboats*, Cardiff, England, 2012. Proceedings of the 5th International Robotic Sailing Conference.
- [5] Marcus Greiff and Anders Robertsson. Optimization based motion planning with obstacles and priorities. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 50:11670–11676, 2017.
- [6] Andouglas Gonçalves da Silva Silva Junior, Davi Henrique dos Santos, Alvaro Pinto Fernandes de Negreiros, João Moreno Vilas Boas de Souza Silva, and Luiz Marcos Garcia Gonçalves. High-level path planning for an autonomous sailboat robot using q-learning. *Sensors*, 20(6):1550, March 2020.

Appendices

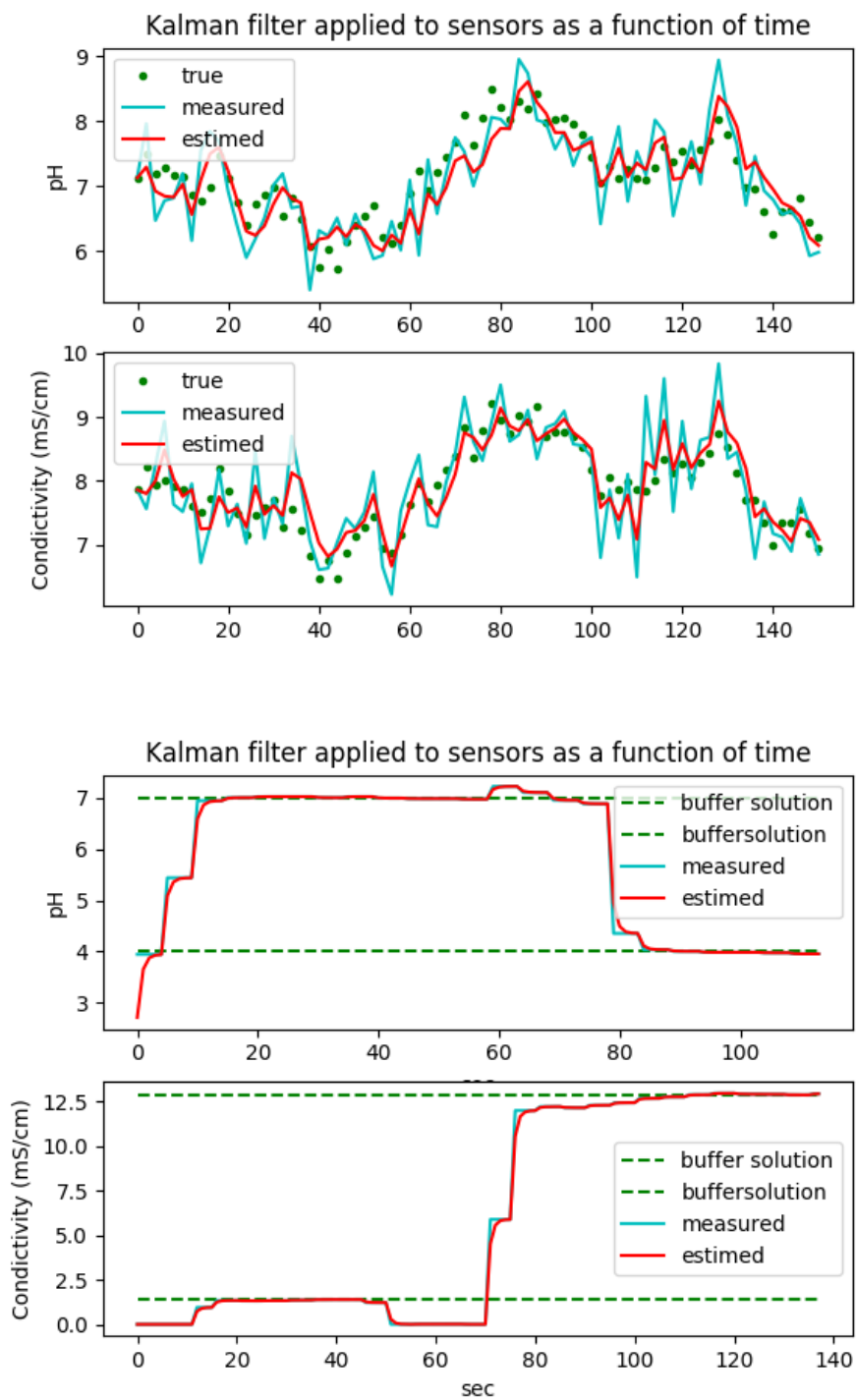
A Complete BLE Attribute Protocol (GAAT) table of the Arduino Bluno

```
catamaran@catamaran:/usr/local/lib/python3.6/dist-packages/bluepy$ python3 btle.py C8:DF:84:24:27:F6
Connecting to: C8:DF:84:24:27:F6, address type: public
Service <uuid=Generic Access handleStart=1 handleEnd=11> :
  Characteristic <Device Name>, hnd=0x2, supports READ
  -> b'Bluno'
  Characteristic <Appearance>, hnd=0x4, supports READ
  -> b'\x00\x00'
  Characteristic <Peripheral Privacy Flag>, hnd=0x6, supports READ WRITE
  -> b'\x00'
  Characteristic <Reconnection Address>, hnd=0x8, supports WRITE
  Characteristic <Peripheral Preferred Connection Parameters>, hnd=0xa, supports READ
  -> b'P\x00\xa0\x00\x00\x00\xe8\x03'
Service <uuid=Generic Attribute handleStart=12 handleEnd=15> :
  Characteristic <Service Changed>, hnd=0xd, supports INDICATE
Service <uuid=Device Information handleStart=16 handleEnd=34> :
  Characteristic <System ID>, hnd=0x11, supports READ
  -> b"\xf6'\x00\x00\x84\xdf\xc8"
  Characteristic <Model Number String>, hnd=0x13, supports READ
  -> b'DF Bluno'
  Characteristic <Serial Number String>, hnd=0x15, supports READ
  -> b'0123456789'
  Characteristic <Firmware Revision String>, hnd=0x17, supports READ
  -> b'FW V1.97'
  Characteristic <Hardware Revision String>, hnd=0x19, supports READ
  -> b'HW V1.7'
  Characteristic <Software Revision String>, hnd=0x1b, supports READ
  -> b'SW V1.97'
  Characteristic <Manufacturer Name String>, hnd=0x1d, supports READ
  -> b'DFRobot'
  Characteristic <IEEE 11073-20601 Regulatory Certification Data List>, hnd=0x1f, supports READ
  -> b'\xfe\x00experimental'
  Characteristic <PnP ID>, hnd=0x21, supports READ
  -> b'\x01\r\x00\x00\x00\x10\x01'
Service <uuid=dfb0 handleStart=35 handleEnd=65535> :
  Characteristic <dfb1>, hnd=0x24, supports READ WRITE NO RESPONSE WRITE NOTIFY
  -> b'\x01'
  Characteristic <dfb2>, hnd=0x27, supports READ WRITE NO RESPONSE WRITE NOTIFY
  -> b'\x02'
```

B Complete Arduino class diagram



C Kalman filter: theoretical and experimental results



D Modeling equations and parameters

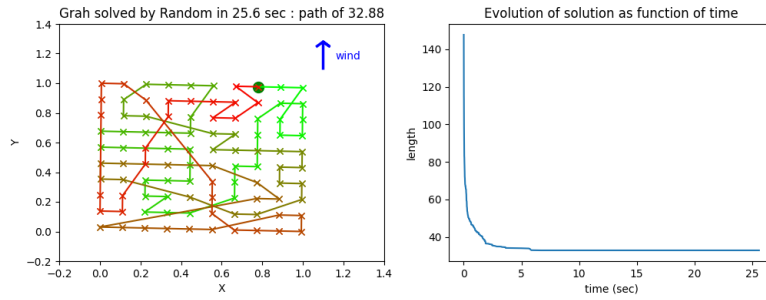
Geometrical parameters linked to the shape of the sailboat

	value	unit	parameter
p1	0.03	—	drift coefficient
p2	40	kgs	tangential friction
p3	6000	kg.m	angular friction
p4	200	kgs	sail lift
p5	1500	kgs	rudder lift
p6	0.5	m	distance to sail CoE
p7	0.5	m	distance to mast
p8	2	m	distance to rudder
p9	300	kg	mass of boat
p10	400	kg.m2	moment of inertia
p11	0.2	—	rudder break coefficient

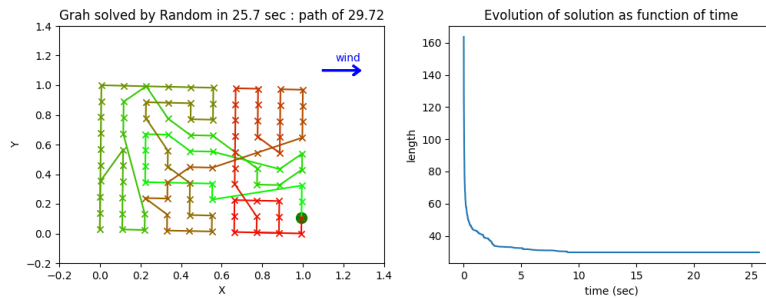
Evolution equations

$$\begin{cases} \dot{x} = v \cdot \cos(\theta) + p_1 \cdot a_{tw} \cdot \cos(\psi_{tw}) \\ \dot{y} = v \cdot \sin(\theta) + p_1 \cdot a_{tw} \cdot \sin(\psi_{tw}) \\ \dot{\theta} = w \\ \dot{v} = \frac{g_s \cdot \sin(\delta_s) - g_r \cdot p_1 \cdot \sin(\delta_r) - p_2 \cdot v^2}{p_9} \\ \dot{w} = \frac{g_s(p_6 - p_7 \cdot \cos(\delta_s)) - g_r \cdot p_8 \cdot \cos(\delta_r) - p_3 \cdot w \cdot v}{p_{10}} \end{cases} \quad (3)$$

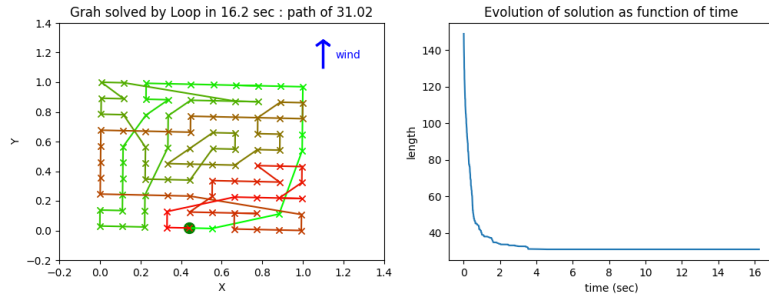
E Effects of wind on path planning



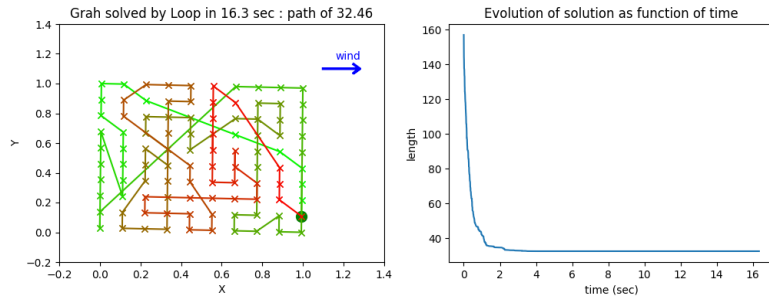
(a) North wind - random strategy



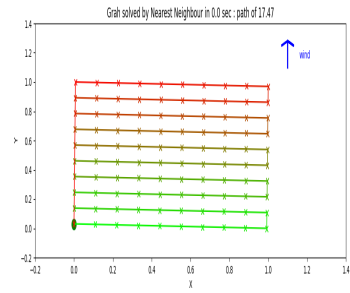
(b) West wind - random strategy



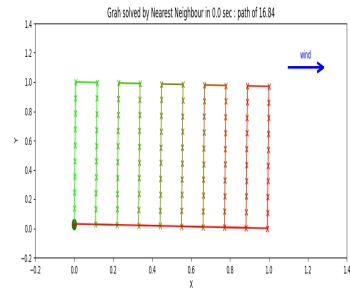
(c) North wind - loop strategy



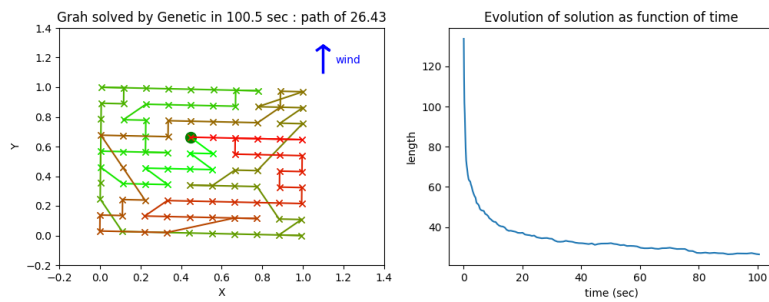
(d) West wind - loop strategy



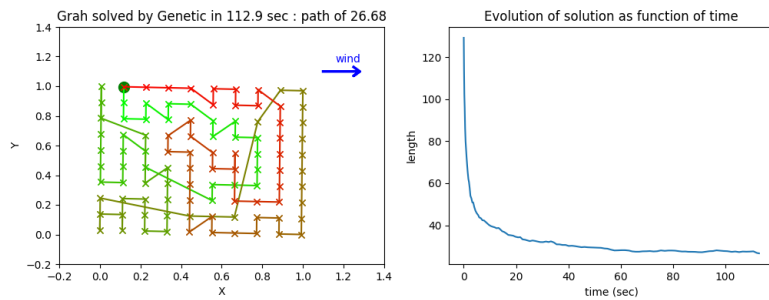
(e) North wind - N N strategy



(f) West wind - N N strategy



(g) North wind - genetic strategy

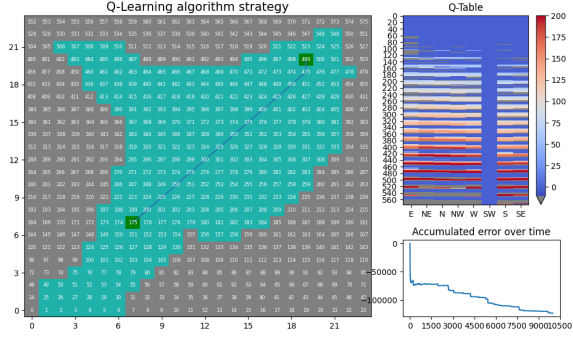


(h) West wind - genetic strategy

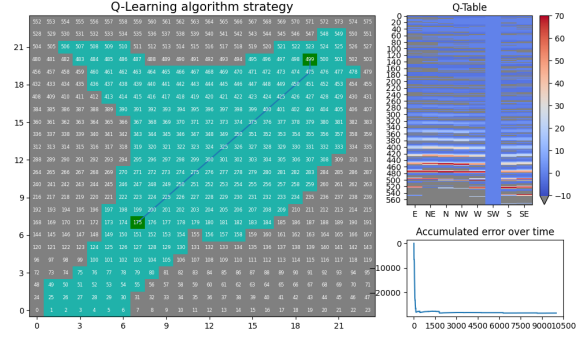
F Comparison of Q-Learning reward matrices

The same hyperparameters and environment, as described in the paper for each scenario, are used. Results from the gradient reward system are on left, from classic reward system on right.

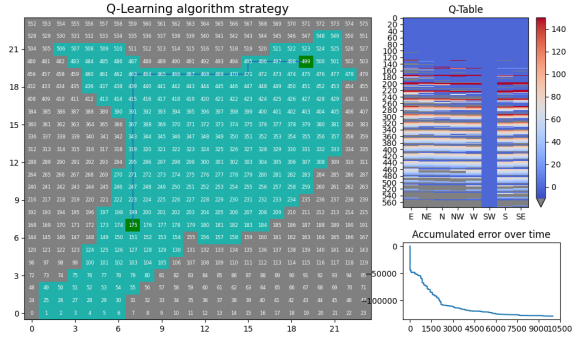
Scenarios 1a/1b (area without obstacles with a fixed wind direction, for different starting and ending points)



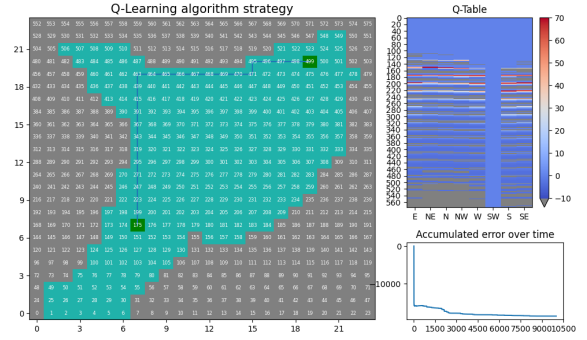
(i) 1a Gradient



(j) 1a Classic



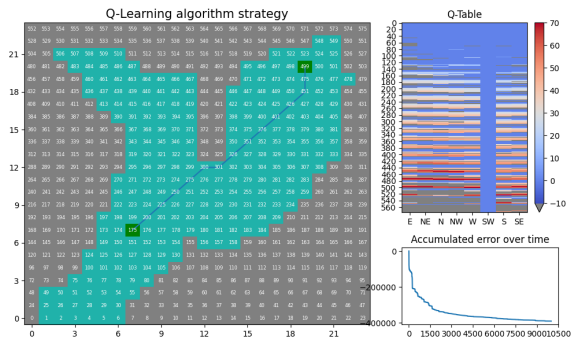
(k) 1b Gradient



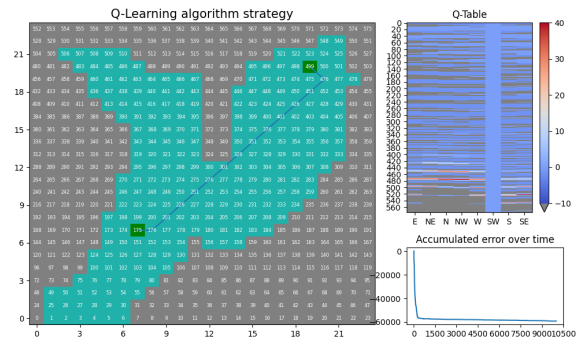
(l) 1b Classic

For scenarios 1a and 1b, the two reward systems converge towards the same solutions, the classic reward system is quicker though.

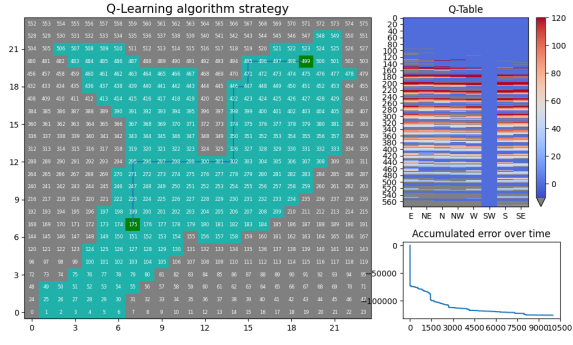
Scenarios 2a/2b (area with obstacles with a fixed wind direction, for different starting and ending points)



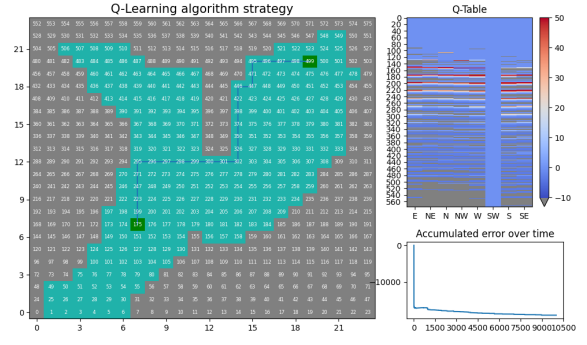
(m) 2a Gradient



(n) 2a Classic



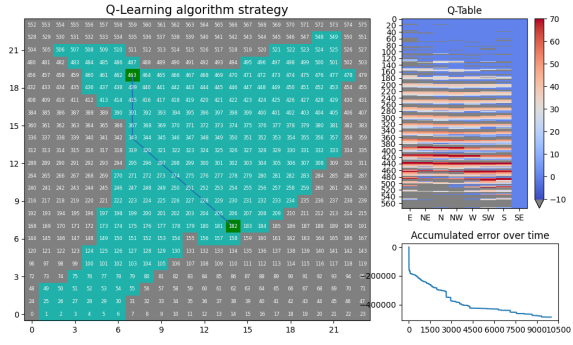
(o) 2b Gradient



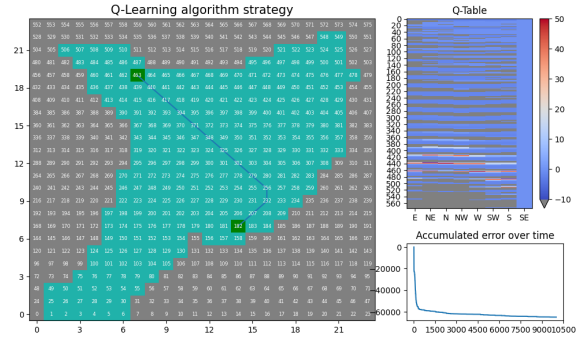
(p) 2b Classic

For scenarios 2a and 2b, with additional obstacles in their zone, the two reward systems nearly converge towards the same solutions. As described in the paper, the convergence takes more time to be reached than in scenarios with no obstacles.

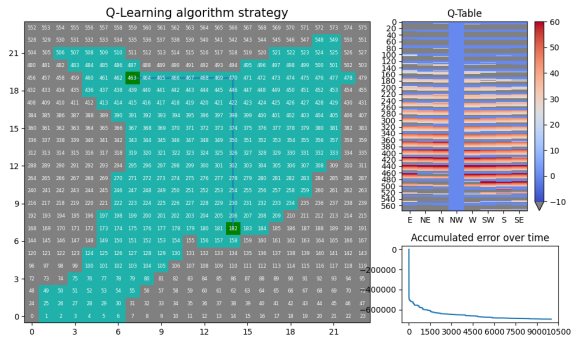
Scenarios 3a/3b/3c (area without obstacles with different wind directions, for same starting and ending points)



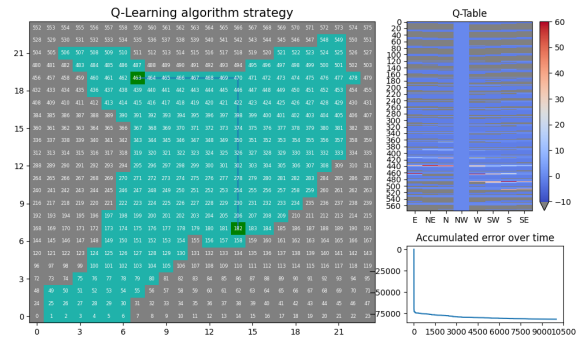
(q) 3a Gradient



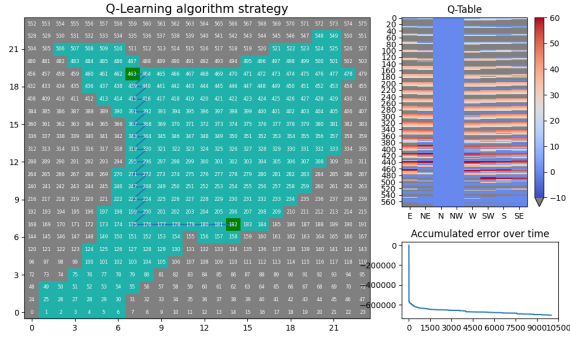
(r) 3a Classic



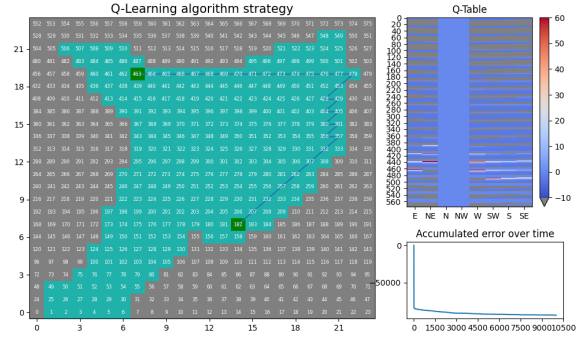
(s) 3b Gradient



(t) 3b Classic



(u) 3c Gradient



(v) 3c Classic

Concerning the last three scenarios, the two reward systems do not converge towards the same solution, but the score of each final path is the same in each case. Maybe, here, the classic reward system (on right) suggests paths with fewer zigzag lines, that could be more easily performed by the boat. However this cannot be generalized as this system could have found as well the same paths as the other one (because of their identical score).

This further comparison has finally led to the same observations as the previous one (Q-Table final appearance, convergence speed). For nearly identical results, the classic reward system could replace the gradient one if convergence speed is a crucial criterion.

G Assessment report



RAPPORT D'EVALUATION ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / HOST ORGANISATION

NOM / Name University of Plymouth

Adresse / Address Drake Circus, Plymouth
Devon PL4 8AA, UK

Tél / Phone (including country and area code) +44 01752 586157

Nom du superviseur / Name of internship supervisor Jian Wan

Fonction / Function lecturer in control systems Engineering

Adresse e-mail / E-mail address jian.wan@plymouth.ac.uk

Nom du stagiaire accueilli / Name of intern

Agathe ARCHET

II - EVALUATION / ASSESSMENT

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
Please attribute a mark from **A (excellent)** to **F (very weak)**.

MISSION / TASK

❖ La mission de départ a-t-elle été remplie ? ✓ A B C D E F
Was the initial contract carried out to your satisfaction?

❖ Manquait-il au stagiaire des connaissances ? ☐ oui/yes ☒ non/no
Was the intern lacking skills?

Si oui, lesquelles ? / If so, which skills? _____

ESPRIT D'EQUIPE / TEAM SPIRIT

❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / If you wish to comment or make a suggestion, please do so here not available for observation
due to work at home

COMPORTEMENT AU TRAVAIL / BEHAVIOUR TOWARDS WORK

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

☒ A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

INITIATIVE – AUTONOMIE / INITIATIVE – AUTONOMY

Le stagiaire s'est-il rapidement adapté à de nouvelles situations ?

A B C D E F

(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations?

☒ A B C D E F

(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

CULTUREL – COMMUNICATION / CULTURAL – COMMUNICATION

Le stagiaire était-il ouvert, d'une manière générale, à la communication ?

☒ A B C D E F

Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

OPINION GLOBALE / OVERALL ASSESSMENT

❖ La valeur technique du stagiaire était :

☒ A B C D E F

Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / FUTURE PARTNERSHIP

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☒ oui/yes

☐ non/no

Fait à _____, le _____
In Plymouth, on 09/09/2020

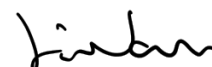
Signature Entreprise

Company stamp

not available due
to lockdown

Signature stagiaire

Intern's signature



Merci pour votre coopération

We thank you very much for your cooperation