STATUS REPORT

Surveillance Of The Bay Of Biscay

Written by :

Elouan Autret
Maxime Bouyssou
Alice Danckaers
Raphael Finkelstein
Pierre Jacquot
Eric Mourre
Benoit Raymond
Thomas Boulier
Philippe Chuzel
David Duverger
Sylvain Hunault
Mael Le Gallic
Thiago Oliveira
Khadimoullah Vencatasamy

Under the supervision of :

Luc Jaulin
Benoit Zerr

ENSTA Bretagne

Option Systèmes Perception Information Décision

Contents

Acknowledgement **4**

1 Introduction **5**

2 Secure zone processing **6**
2.1 Introduction to interval analysis 6
2.2 Secure zone estimation with interval analysis 8
2.2.1 Thick Functions 8
2.2.2 Test for Biscay Bay Surveillance 9
2.3 Map generation and origin estimation 15
2.3.1 Origin Determination 16
2.4 Mesh to boxes functions 17
2.5 Erosion functions 18
2.5.1 Trail Modelization 18
2.6 Global Sivia algorithm 21
2.7 SIVIA Algorithm 21
2.8 Results 23

3 Robots regulation **24**
3.1 Regulation 24
3.1.1 Artificial Potentials Field 24
3.1.2 Feedback Linearisation 25
3.2 Evolution of the elliptic trajectory 26
3.3 Conversion of the order to a PWM command 26

4 Implementation with buggy robots **27**
4.1 Hardware 27
4.1.1 Hardware architecture 27
4.2 GPS Sentence Analysis 28
4.3 Structure of the programming code 29
4.4 Results 32

Conclusion **33**

Appendix **34**

A Appendix 1 **35**

B Appendix 2 **36**

C Appendix 3 **37**

Bibliography **37**

Acknowledgement

Chapitre 1

Introduction

The goal of this project is to ensure the security of the Bay of Biscay by detecting any intruder entering this maritime area. A group of robots, equipped with GPS, are being used thanks to specific algorythms and a particular monitoring strategy.

In order to proceed with the project, the following working hypotheses are considered :

— The simulation can be done in a two-dimensional world ;
— Intruders will have a constant given velocity throughout the simulation ;
— Every surveillance robot has a detection range of d_i within which an intruder is necessarily detected ;
— The safe zones are regarded as a group of rectangles where the intruder can not be.

Two deliverables will be delivered :

— The first one is a theoretical simulation written to a large extent in Python 3.0 which will allow to determine whether or not the used algorythms are relevant ;
— The second one is a practical simulation using some buggies robots available in the robotics group of ENSTA Bretagne.

Chapitre 2

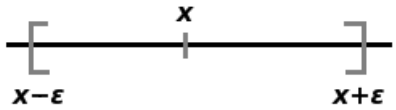
Secure zone processing

2.1 Introduction to interval analysis

David DUVERGER

For this project, we have chosen the intervals calculation approach which guarantees solutions in a given interval of uncertainty.

In the Intervals theory, a known value is replaced by an interval with a certain uncertainty in which it is sure that this measure is include. So, for a known value x, the corresponding interval is $[x-\epsilon, x+\epsilon]$.



The goal of this method is to give the percentage of certainty which allows to obtain reliable and robust results. In tow dimensions, intervals are represented by boxes. Here is an example of the resolution by Intervals. It is the result of S1|S2 with :

$$\begin{aligned}\mathbb{S}_1 &= \left\{(x,y)\in\mathbb{R}^2 \mid (x-1)^2+(y-2)^2 \in [4,5]^2\right\}\\ \mathbb{S}_2 &= \left\{(x,y)\in\mathbb{R}^2 \mid (x-2)^2+(y-5)^2 \in [5,6]^2\right\}.\end{aligned}$$

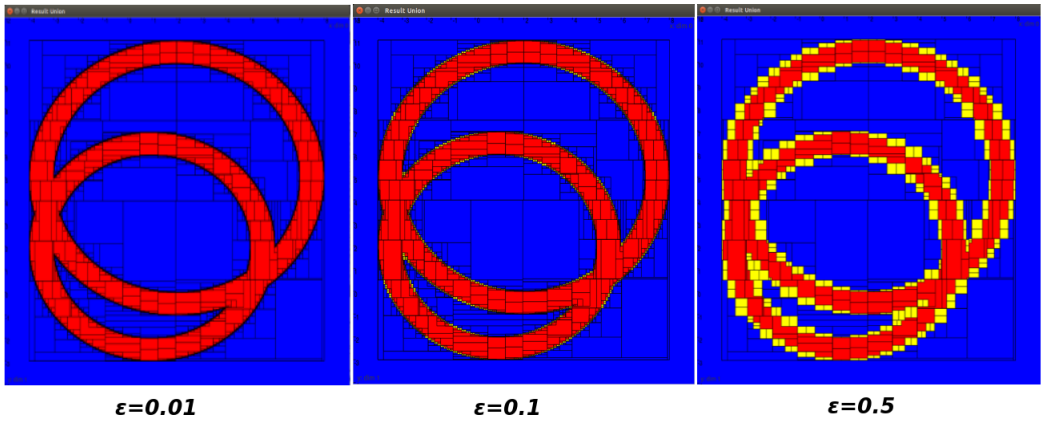


FIGURE 2.1 – Resolution of S1 U S2 for different precision with intervals calculation

If a box frames a solution, it is cut in two parts and the calculation is reiterated on these two boxes which allow to discriminate one of them and to reduce solutions. However, this operation take lot of time because the number of boxes increase of 2n each time. That is why, the contractors are used, they allow to refine the boxes at the intervals which include strictly the solution before to cut them. This method allow to have an important gain of time during the simulation.

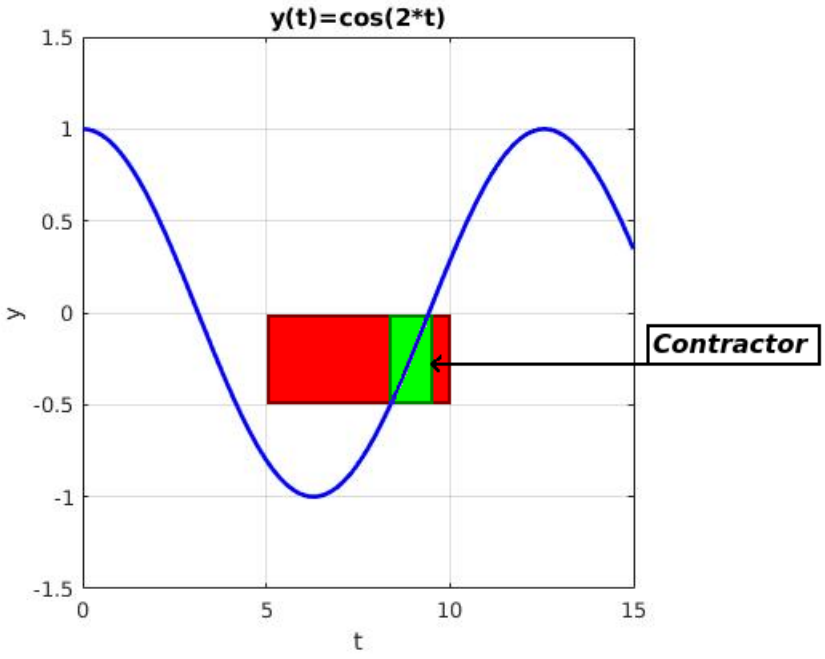


FIGURE 2.2 – Example of the utilisation of contractor

For the example, the signal cos(2*t) in blue is used. The box in red corresponds to the normal box and the green box corresponds to the contractor. With this example, we can easily see that contractors improve the simulation in terms of time.

For the simulation of the Gascogne project, we used a SIVIA function which make the paving for the representation of the France, robots control and their erosion.

2.2 Secure zone estimation with interval analysis

Elouan AUTRET

2.2.1 Thick Functions

A thick function is a function of $\mathbb{R}^n$ in $\mathbb{R}^p$ that associate to an element x $\in \mathbb{R}^n$ a convex element of $\mathbb{R}^p$ where there can have intervals as parameters of the function.

For example, consider the function f :

$\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}$

$\mathbf{x} \rightarrow \mathbf{x} + [1, 2]$

If the width of x is null (ex : [1,1]) the width of f(x) will be strictly greater than zero ([2,3]) (see figure 2.3).

A thick function can be consider as an interval of classical function with the same principal properties like an lower bound and an upper bound, [f] is equal to [f^-,f^+] where f^- is the lower bound function and f^+ is the upper one. See the example for a function from $\mathbb{R}$ in$\mathbb{R}$:

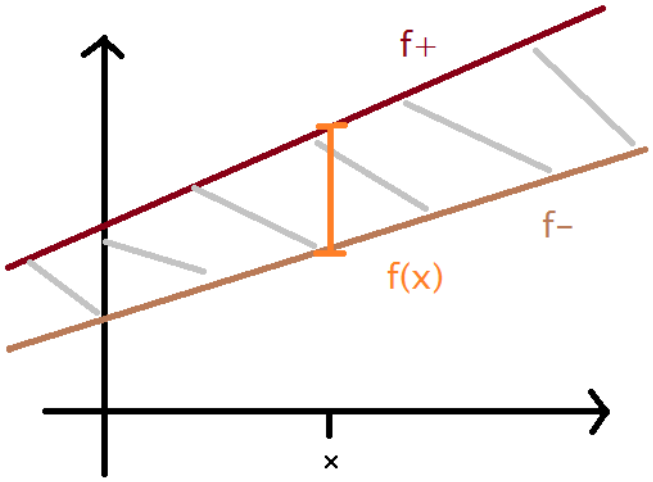


FIGURE 2.3 – Example of a thick function $\mathbb{R} \rightarrow \mathbb{R}$.

Those thick functions allow, for the surveillance problem, the modelling of the uncertainty of the position of the boat that will watch the bay.
Indeed, the position can be given by a GPS but the result is not absolute, the real position can be off by a few meters, for example if [1,2] is the position of the boat and 10 is the range of detection by the boat, then the set of the secure zone without uncertainty would be :

$$\mathbf{S} = \{x \in \mathbf{R}^2 | (x(1) - 1)^2 + (x(2) - 2)^2 \leq 100\} \tag{2.1}$$

But if we add an uncertainty of 0.5 to the position then the set become :

$$\mathbf{S} = \{x \in \mathbf{R}^2 | (x(1) - [0.5, 1.5])^2 + (x(2) - [1.5, 2.5])^2 \leq 100\} \tag{2.2}$$

This set can easily be found with interval analysis using the ibex library with a separator for the precedent equation (2.1), then by proceeding with the SILVIA algorithm and associating the code with VIBES (a viewer tool) the zones reached by the boat can be visualized :

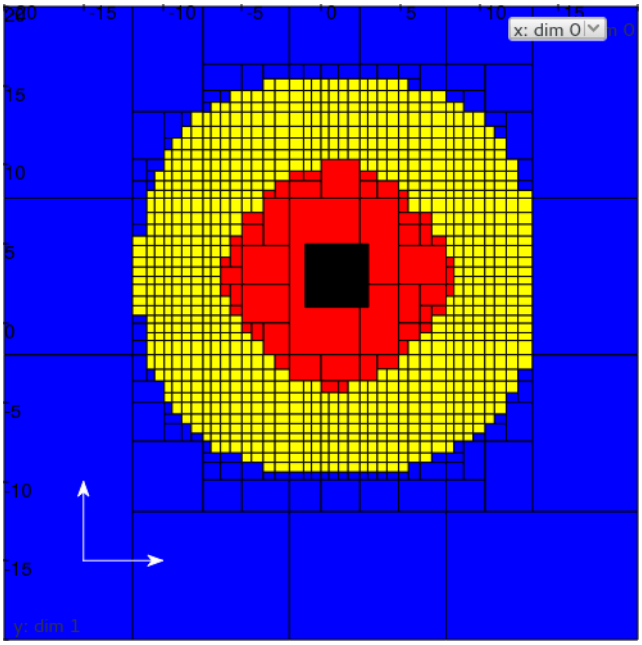


FIGURE 2.4 – Zone secured by one boat in red, yellow zone is uncertain,blue is not secured,the black square are the possible position of the boat.

The figure 2.4 shows the zone covered by the surveillance boat, in red the zone is certain to be covered, in yellow it is not sure it depends on the real place of the boat in the black square or uncertainty.

2.2.2 Test for Biscay Bay Surveillance

For all this section we consider that for all $\mathbf{X} \in \mathbb{R}^2$, $\mathbf{X}$ is included in the bay of Biscay (the set $\mathbb{G}$) in order to facilitate the comprehension.
The figure 2.4 shows that the zone covered by the boats can be found but it also point out that the computing of the zone is not efficient, the uncertain zone cannot be classify as out or in the secured zone so the SIVIA algorithm cut the boxes to their minimal size even if it is possible to know their status of uncertainty thus proceeding to more computation than needed. A new test is required instead of a simple separator to allow a faster and cleaner computation of the secured zone. The test can be found in the algorithm 1 which take a box (an element of $\mathbb{R}^2$)) and determine its status, the state can vary between six status where for now the border regroup more or less three status as it can be seen in the following table :

Symbol	In Algorithm	Meaning
0	OUT	Box is not in the Secure Zone
1	IN	Box is in the secure zone
?	UNKNOWN	Box is at the border of the secure zone
[0, ?]	UNKNOWN	Box is on the external border of the covert zone
[?,1]	UNKNOWN	Box is on the internal border of the covert zone
[0,1]	MAYBE	Box is in the uncertainty zone

The test 1 need to invert a thick set created by the function in use, here it is the euclidean norm for interval, the bounds of the thick function are used in algorithm 1 to compute the set inversion :

$$f : \left| \begin{array}{ccc} \mathbb{R}^2 & \longrightarrow & \mathbb{R} \\ X & \longmapsto & \|X - m\|^2 \end{array} \right.$$

Where :

$$f^-(X) = (X[0] - m[0])^2 + (X[1] - m[1])^2$$

$$f^+(X) = max((X[0]-m[0]^-)^2, (X[0]-m[0]^+)^2) + max((X[1]-m[1]^-)^2, (X[1]-m[1]^+)^2)$$

See paragraph 4.5 in [**?**] for the computation of the bounds.

Algorithm 1 Is $\mathbf{X} \subseteq \mathbb{S}_m$, $\mathbb{S}_m$ = Secured Zone by boat m and $\mathbf{X} \in \mathbb{R}^2$

Require: X,$range$ (reach of boat),m (position of boat)
$f : \mathbb{R}^2 \to \mathbb{R}$
$x \to \|X - m\|^2$
$Xm \leftarrow f^-(X)$
$Xp \leftarrow f^+(X)$
$Xub \leftarrow Xp|Xm$
if $Xub \cap [0, \text{range}^2] = \emptyset$ **then**
  return OUT
else if $Xub \subseteq [0, \text{range}^2]$ **then**
  return IN
else
  if $\text{range}^2 - Xp^- < 0$ **then**
    if $Xm - \text{range}^2 \subseteq [-\infty, 0]$ **then**
      return MAYBE
    else
      return UNKNOWN (later :UNKNOWN2)
    end if
  else
    return UNKNOWN
  end if
end if

The solution of the zone covered by the boat resemble to the figure 2.5, and by seeing the size of the boxes (there are less boxes) in the uncertain zone it is sure that the computation was efficient :

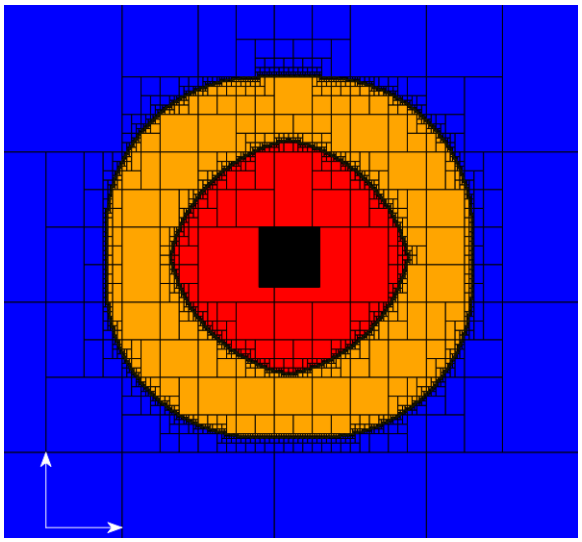


FIGURE 2.5 – Zone secured by one boat in red, orange zone is uncertain, yellow is the border and blue is not secured.

In order to easily apprehend the algorithm 1 the figures 2.6, 2.7, 2.10, 2.8 and 2.9 represent in a graphics way the upper and lower bound of the function f :

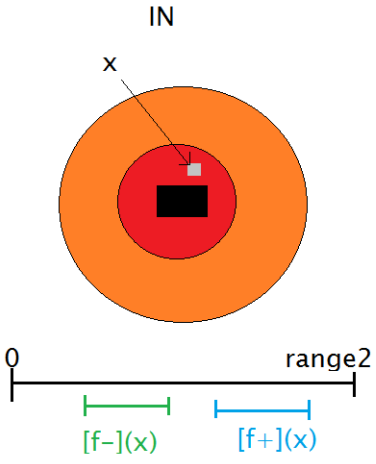


FIGURE 2.6 – Representation of the interval bound of an IN box.

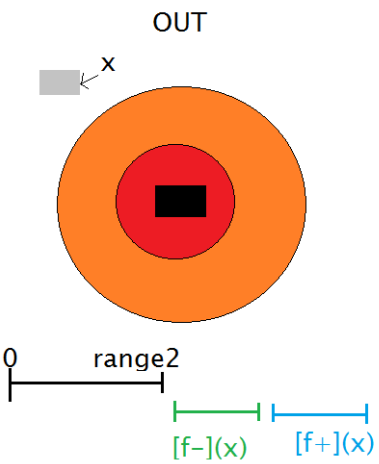


FIGURE 2.7 – Representation of the interval bound of an OUT box.

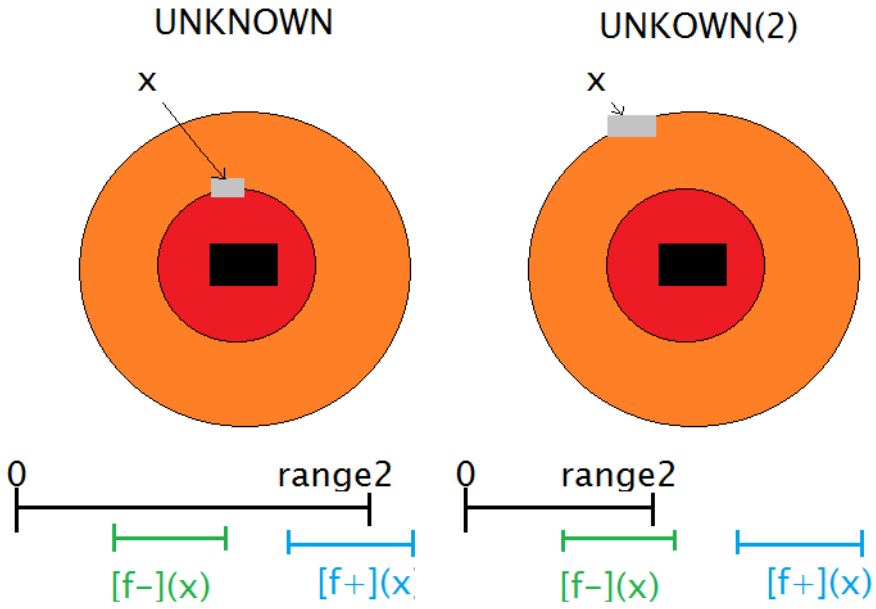


FIGURE 2.8 – Representation of the interval bound of a box at the border between the secured zone and the uncertain zone.

FIGURE 2.9 – Representation of the interval bound of a box at the border between the Out zone and the uncertain zone.

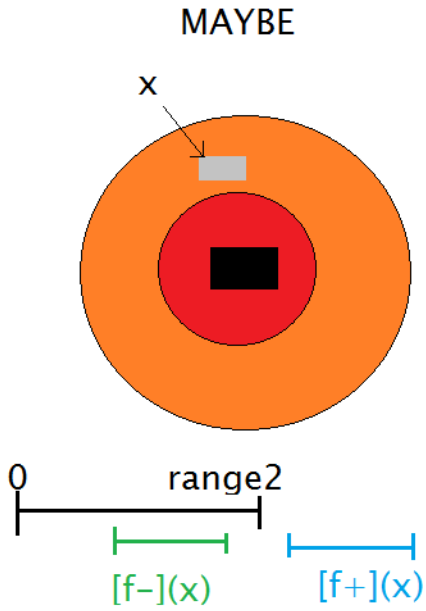


FIGURE 2.10 – Representation of the interval bound of a box in the uncertain zone.

The intervals representing $[f]^{-}(X)$ and $[f]^{+}(X)$ are draw arbitrary in size and position, for example $[f]^{-}(X)$ and $[f]^{+}(X)$ can overlap (except in figure 2.10) what is important is the relative position of their bounds compared to the bounds of [0,range^2].

Now that the algorithm is working for on boat, other can be added :

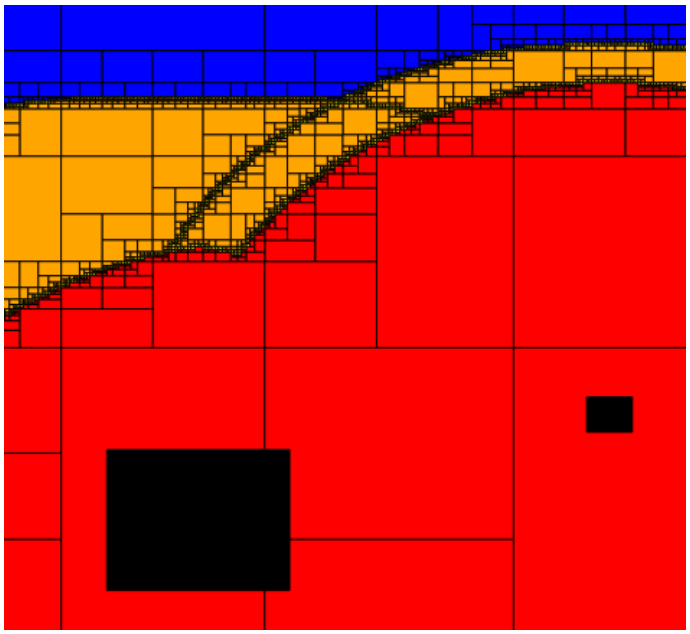


FIGURE 2.11 – Zone secured by two boat in red, orange zone is uncertain, yellow is the border.

The table shown earlier was not completely correct, as it was omitting the overlapping of the different boats as seen in the figure 2.11, the external border of the zone covered by on boat is overlapping the uncertain zone of a different boat.

In order to avoid those wrong and useless computations, the algorithm 1 need to take into account the fact that an external border for a boat can be in an uncertain zone of an another boat, this change the correspondence table to :

Symbol	In Algorithm	Meaning
0	OUT	Box is not in the Secure Zone
1	IN	Box is in the secure zone
?	UNKNOWN	Box is at the border of the secure zone
[0, ?]	UNKNOWN2	Box is on the external border of the covert zone
[?,1]	UNKNOWN	Box is on the internal border of the covert zone
[0,1]	MAYBE	Box is in the uncertainty zone

Now the algorithm 1 need to update this is done by changing the return from UNKNOWN to UNKNOWN2 in operand 13.
But the result seen in figure 2.11 need to fusion result of algorithm 1 between the different boat, this can be done by evaluating the result two by two boats, which correspond to the Truth table that follow :

Test1/Test2	0	1	?	[0, ?]	[?,1]	[0,1]
0	0	1	?	[0, ?]	[?,1]	[0,1]
1		1	1	1	1	1
?			?	?	[?,1]	[?,1]
[0, ?]				[0, ?]	[?,1]	[0,1]
[?,1]					[?,1]	[?,1]
[0,1]						[0,1]

This truth table is translated in an algorithm(see 2).

Algorithm 2 Is $\mathbf{X} \subseteq \mathbb{S}$, $\mathbb{S}$ = Secure Zone and $\mathbf{X} \in \mathbb{R}^2$

Require: X,$range$ (reach of a boat),M (list of position of boats)

$R \leftarrow \{$Algorithme 1(X,range,m)$\}_{m \in M}$
if IN $\subset R$ **then**
    return IN
else if UNKNOWN $\subset R$ **then**
    return UNKNOWN
else if MAYBE $\subset R$ **then**
    return MAYBE
else if $\forall r \in R$, $r =$ OUT **then**
    return OUT
else
    return UNKNOWN
end if

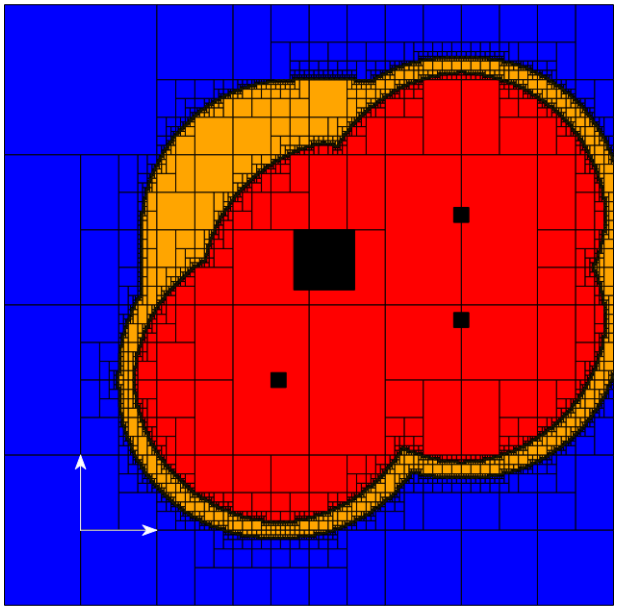


FIGURE 2.12 – Zone secured by four boat in red, orange zone is uncertain,blue is not secured.

Those last changes make the computation of the zone more efficient and correct, in figure 2.12 there is no more overlapping of the external border.

If the uncertain zone is not needed the computation of the algorithm SIVIA can be accelerated by ignoring it. This can be done by changing the return value in the operand 13 in the algorithm 1 from UNKNOWN2 to OUT and the return value in the operand 11 from MAYBE to OUT (the second change will not improve efficiency but just place the boxes in the right set), the result can be observed in figure 2.13.

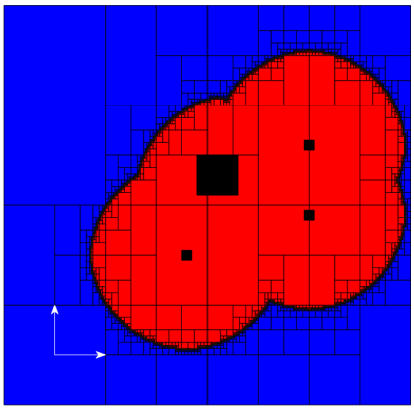


FIGURE 2.13 – Zone secured by four boat computed with algorithm 2 and 1.

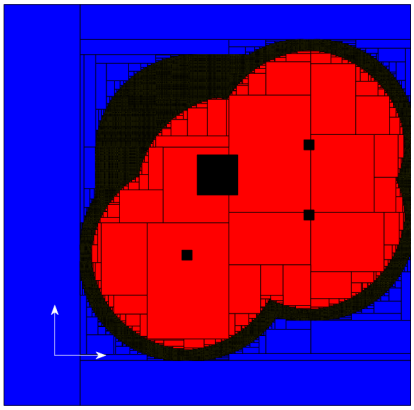


FIGURE 2.14 – Zone secured by four boat computed with separators

The result in figure 2.13 may also be obtain with the combination of separators (see figure 2.14 but the result will be longer to get as it have to cut through the uncertain zone even if it is time consuming (and with no information gain). To demonstrate the waste of time done by the separators the figure 2.14 has taken thirty six times the time taken to compute the figure 2.13 (fastest computation times).

2.3 Map generation and origin estimation

Khadimoullah Vencatasamy

We want to simulate on a map and we want to generate the map easily. So the easiest way we found was to take a screen on Google earth where we know the scale. Our objective is to transform this map into a binary image. To do so we choose to make a color filter to isolate the ground from the sea. We thought to use a contour filter but due to the many bumps present on the satellite view sea it was successful.

So we have a script which take a satellite view as input, when you run this script you have to windows : one with some trackbar which you can slide to filter the color you want and apply a closing filter to erase the last parasite pixels. The second window show the output image. Press 's' to save the output image.

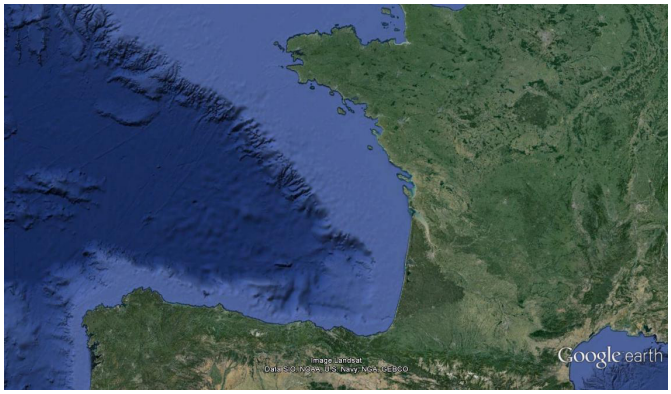


FIGURE 2.15 – Input Image

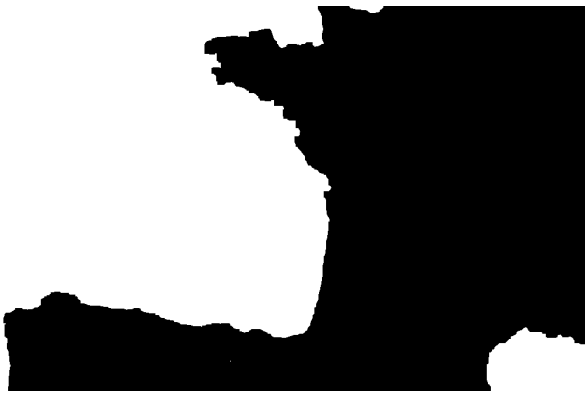


FIGURE 2.16 – Output Image

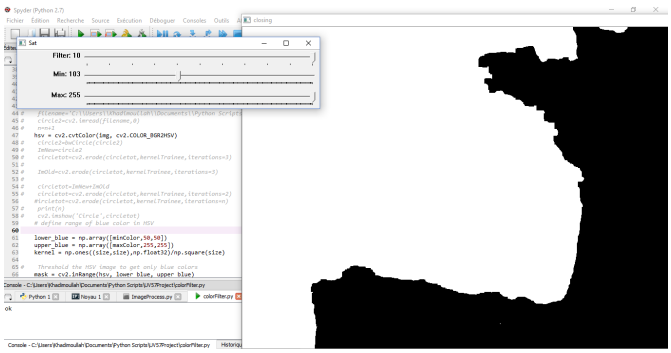


FIGURE 2.17 – Interface to binarize the satellite image

This is this image which is use to generate the paving in ImageToBoxes script.

2.3.1 Origin Determination

In our problem we set the origin of the map to the point 45°N 4°W so we need to find the (i0,j0) which are the coordinates of the origin.

To achieve this we need the latitude and the longitude of the top left corner of our binary map. We will note X Y for the coordinate in pixel and LAT LONG for the coordinate in WPs.

First let's determine the δX and δY :

In our case we have chosen (45°N,4°W) as origin which corresponds to the point (410 400) in pixel coordinate.

2.4 Mesh to boxes functions

Maël LE GALLIC

In order to define the *Bay of Biscay*, $\mathbb{G}$, we have to implement a SIVIA test that distinguish the earth from the sea. To deals with it, we realised a procedure which returns the sub-pavings of an input binary image. First of all, we compute the integral image of the binary image. The integral image (or summed area table) is a data structure which permit to quickly and efficiently generate the sum of pixel values in a rectangular subset of an image. The value at any pixel (i, j) in the integral image correspond to the sum of all the pixels above and to the left of (i, j). To compute the sum of pixel values of a rectangular subset of the image only four pixel values of the integral are required. Hence, the calculation time is constant and independent of the size of the rectangular area. This sum of i(i,j) over the rectangle spanned by A, B,C and D is :

equation

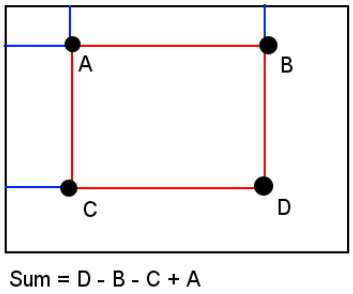


FIGURE 2.18 – Description of computing a sum in an integral image

Then, we use this integral image in our SIVIA test to say if the tested box is inside the 0-bit area, the 1-bit area or overlaps the two areas of the binary image. The procedure of the test is described below :

— We calculate the indexes of the four pixels corresponding to the vertices of the box.
— We compute the number of pixel in the box : nbsPixel
— By using the integal image, we compute the sum of pixel values in the box : sumPixel

An example of the application of the SIVIA algorithm with the ImageToBoxes test is

shown below. In input we have a binary image of the Biscay Bay and in output the resulting sub-pavings.

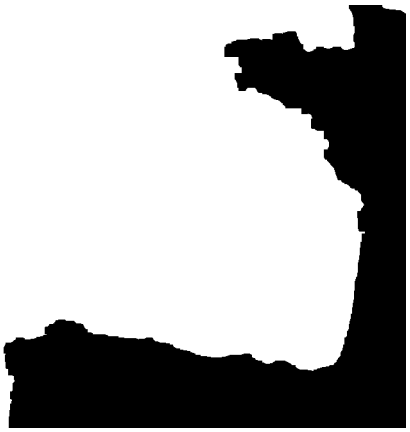


FIGURE 2.19 – Binary image of the Biscay Bay

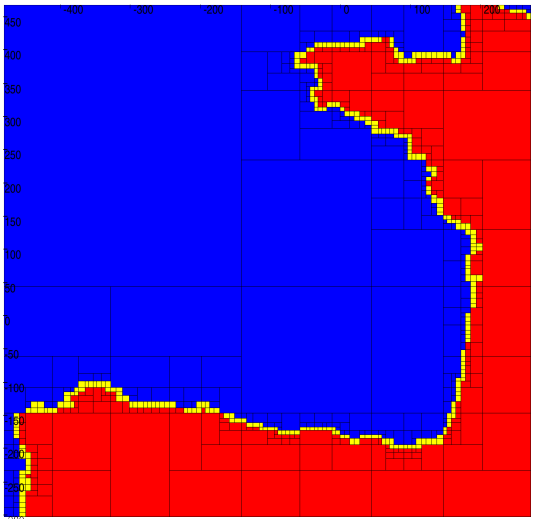


FIGURE 2.20 – Resulting sub-pavings of the Biscay Bay

2.5 Erosion functions

Khadimoullah Vencatasamy

2.5.1 Trail Modelization

We have define the secure zone with this equation :

$$\mathbb{X}(t) = \mathbb{G} \cap \mathbb{F}_{\delta}(\mathbb{X}(t-\delta)) \cap \bigcap_{i\leq}((g_(a_i))^{-}1(d_i(t),\infty))) \tag{2.3}$$

Where $\mathbb{F}_{\delta}(\mathbb{X}(t-\delta))$ represents the past of the secure zone. In fact, we can associtae that to the trail of our robot. To compute the trail we have got to solutions. The first one was to stock the positions of our robot and then generate virtual robot with a smaller field of view. But with this solution we need to compute a lot ressources in interval analysis. So the second idea was to apply an erosion on an image. So we do not have to compute interval analysis. Then we give this new image with the trail to the script ImageToBoxes saving a lot of time of calculation.

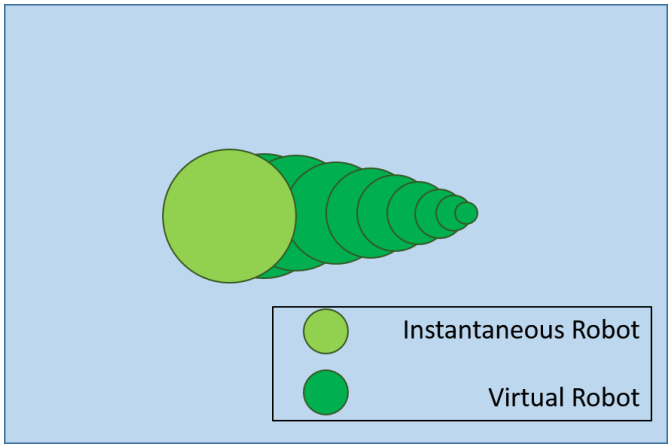


FIGURE 2.22 – Trail Implementation with erosion

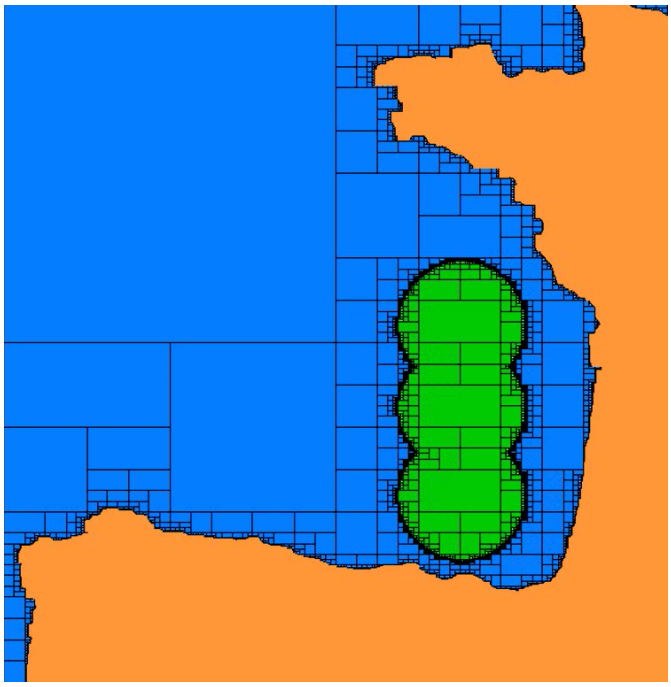


FIGURE 2.21 – Robot without

Finally, the erosion allow us to conclude if our algorithm is efficient. In fact, when the swarm covers the Bay of Biscay, even if a little zone is not covered the erosion will expand this area.

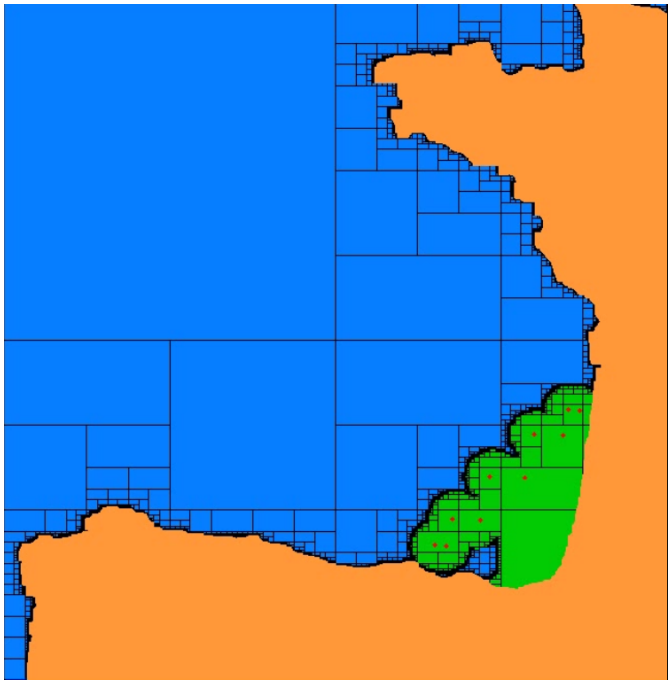


FIGURE 2.24 – The uncovered area is expanded due to the erosion

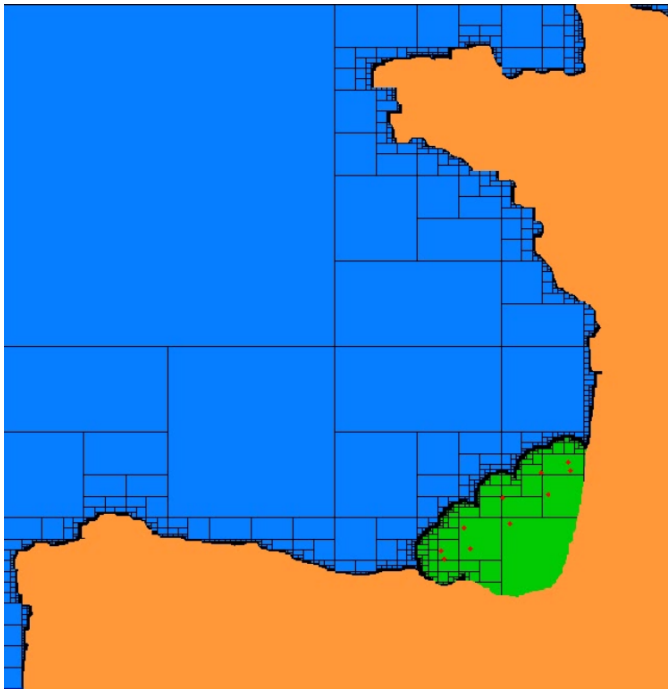


FIGURE 2.23 – The swarm misses a little area

2.6 Global Sivia algorithm

2.7 SIVIA Algorithm

David DUVERGER

We will present here the fusion of the paving of the Gascogne Golf, the robots and their erosion. So, it is the resolution of the following mathematical formula :

$$X(t) = G \cap F_{\delta}(X(t-\delta)) \cap g^{-1}([di, \infty])$$

Algorithm 3 SIVIA algorythm

Inputs

1: Area $X0$
2: Object $France$
3: Object $Erosion$
4: Object $Robots$
5: Float $Precision$

Output

1: Boxes $FranceURobotsUErosion$
2: Boxes $RobotsUErosion$
3: Boxes Sea

Initialization

1: $stack \leftarrow X0$
2: $BoxesFranceURobotsUErosion \leftarrow []$
3: $BoxesRobotsUErosion \leftarrow []$
4: $BoxesSea \leftarrow []$

Algorithm 4 SIVIA algorythm (continued)

while len of stack >0 **do**

$X \leftarrow Interval\,Vector\,of\ stack$
$FranceTest \leftarrow France.function(X)$
$ErosionTest \leftarrow Erosion.function(X)$
$RobotsTest \leftarrow Robots.function(X)$

//If we are in the France, robots or erosion :
if FranceTest==IBOOL.IN or ErosionTest == IBOOL.IN or RobotsTest == IBOOL.IN **then**
$Add\ X\ to\ BoxesFranceURobotsUErosion$

//If we are in the robots or erosion :
if RobotsTest == IBOOL.IN or ErosionTest == IBOOL.IN **then**
$Add\ X\ to\ BoxesRobotsUErosion$
end if

//Else if we are outside all :
else if $FranceTest == IBOOL.OUT\ and\ ErosionTest == IBOOL.OUT\ and\ RobotsTest == IBOOL.OUT$ **then**
$Add\ X\ to\ BoxesSea$
else

if Size of X > Precision **then**
$(X1, X2) \leftarrow cut\ \ X$
$Add\ \ X1\ \ in\ \ stack$
$Add\ \ X2\ \ in\ \ stack$
end if
end if

end while

return BoxesFranceURobotsUErosion,BoxesRobotsUErosion,BoxesSea

With the help of this algorythm we can have a paving of the intersection between the Gascogne Golf, the robots and their erosion. We arrive to obtain the following result :

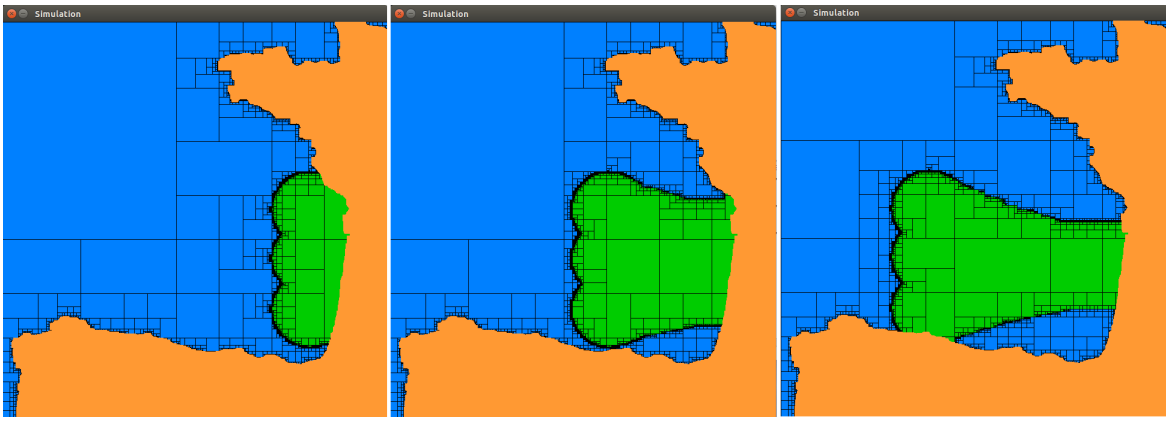


FIGURE 2.25 – Paving of the intersection between the Gascogne Golf, the robots and their erosion

2.8 Results

Distribution of the robots

At first we thought about distributing the robots with a regular spacing along the curvilinear abscissa of the ellipse. This method being too difficult in terms of mathematical theory as well as algorithmical means, this solution was abandoned. In fact, we should have approximated the Wallis' integrals to a limited development which is quite prickly. This would have implied a long programming time and a processing time too lengthy for a simple feasibility study.

This is why we decided to forget this method in benefit for an easier one which involves angular delays.

Thus, each robot follows the previous one with a given angular delay depending on how many robots there are covering the ellipse. This implies that the nearer the robots are to the apogees, the nearer they will be with one another and on the contrary, the nearer they are to the perigees, the farer they are with one another. However, the non linearity in terms of distance between every robots does not impede the conservation of the secure zone. As a matter of fact the robots are faster when they are near the perigee. Therefore the streaks are bigger and so fill in the gaps between the robots. This way we can guarantee the continuity of the secure zone the robots are watching over.

Chapitre 3

Robots regulation

3.1 Regulation

Two methods have been studied and implemented to regulate the pack of robots. The first approach is a method based on artificial potentials field. This method is robust and easy to debug. It was chosen to regulate the real robots on the testing field because it was easier to implement on the robots and provided good results. The second one is a method of feedback linearisation. This method was chosen for the simulation of the surveillance of the Bay of Biscay because it is a classic and efficient method.

3.1.1 Artificial Potentials Field

Thomas Boulier & Sylvain Hunault

Let p_{robot} be the position of our considered robot, let p_{target} and v_{target} be the position and speed of the target we want the robot to reach. We can consider the robot and the target as two particles of opposite charge, and then compute the potentials field between them. In case of obstacles, we can consider them as particles of same charge than the robot's. The potentials field method calculates the instantaneous speed vector $w(p_{robot}, t)$ the robot needs to reach (or at least follow) the target. To compute that speed, we use the potential V between the robot and the target :

$$V(p_{robot}) = v_{target}^{T}.p_{robot} + \|p_{robot} - p_{target}\|^2$$

And compute the gradient of that potential to find the order $w(p_{robot}, t)$:

$$w(p_{robot}, t) = -grad(V(p_{robot})) = -\frac{dV}{dP}(p)^T$$

So :

$$w(p_{robot}, t) = v_{target} - 2.(p_{robot} - p_{target})$$

For w, we compute the order speed and course :

$$\bar{v} = \|w\|$$

$$\bar{\theta} = tan(\frac{w_y}{w_x})$$

As illustrated in the picture below, a proportional regulation computes the command u_v and u_θ which will be used to control the robot.

$$u_v = Kp_v \cdot (\bar{v} - v_{robot})$$

$$u_\theta = Kp_\theta \cdot (\bar{\theta} - \theta_{robot})$$

Where θ_{robot} is the course of the robot calculated by finite difference because the GPS of the robot do not provide the course.

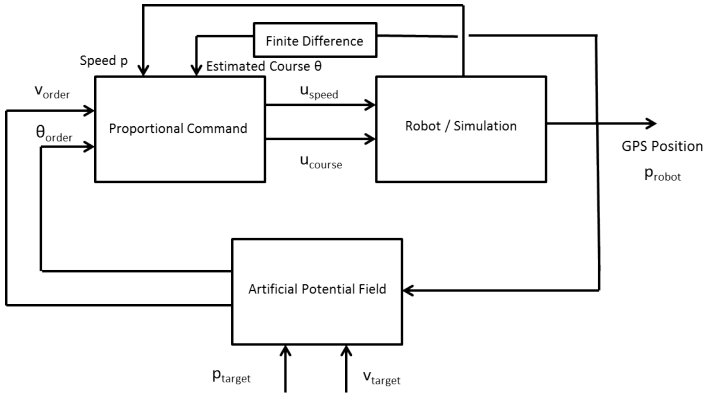


FIGURE 3.1 – Architecture of the potentials field method

u_v and u_θ are saturated to avoid an overloading of the actuators.

This command is then used as the entry of a simple simulator used to test the regulation. The command is used as the acceleration of the robot, and its position at each time is computed through an Euler method. The objective point is on an parametric ellipse moving with time.

The method is easily extended to multiple robots : each one is controlled separately. For each iteration of the loop, the regulation process is applied on each robot. The command is calculated for each one, then the simulation uses it to compute the new position of each one.

u_v and u_θ are saturated to avoid a surcharge of the actuators.

3.1.2 Feedback Linearisation

Alice Danckaers

Feedback linearisation is a classical approach used in controlling nonlinear systems. We transform the nonlinear system 3.1 into an equivalent linear system 3.2, which is of

the form $\ddot{y} = A(x) * u$.

$$\dot{x} = f(x,u) \rightarrow \begin{cases} \dot{x_1} = x_4 * \cos(x_3) \\ \dot{x_2} = x_4 * \sin(x_3) \\ \dot{x_3} = u_1 \\ \dot{x_4} = u_2 \end{cases} \tag{3.1}$$

$$\begin{pmatrix} \ddot{y_1} \\ \ddot{y_2} \end{pmatrix} = \begin{pmatrix} -x_4 * \sin(x_3) & \cos(x_3) \\ x_4 * \cos(x_3) & \sin(x_3) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad \text{with } y = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{3.2}$$

If w is the instruction of the robot has to follow, then we have the expression of the command u as shown in equation 3.3. We can then use it to regulate the robot in an Euler method.

$$u = \begin{pmatrix} -x_4 * \sin(x_3) & \cos(x_3) \\ x_4 * \cos(x_3) & \sin(x_3) \end{pmatrix}^{-1} \begin{pmatrix} (y_1 - w_1) + (\dot{y_1} - \dot{w_1}) - \ddot{w_1} \\ (y_2 - w_2) + (\dot{y_2} - \dot{w_2}) - \ddot{w_2} \end{pmatrix} \tag{3.3}$$

3.2 Evolution of the elliptic trajectory

3.3 Conversion of the order to a PWM command

Thomas Boulier & Sylvain Hunault

The regulation process returns the two values (u_v,u_θ) as ouput. But those values are in "natural" unit, which is acceptable for a simulation but not for the regulation of actual robots. At the end of the regulation, the command is saturated, so the range of u_v is [0,1] , while the range of u_θ is [-10,10]. The command needs to be transformed in an acceptable value for the robots. Indeed, the command taken by the robots is composed of Pulse Width Modultion (PWM) values in the range [1000,2000]. Because we need to keep the robots always moving but not too fast. Moreover we do not want to allow the robots to have a null speed or worst, a negative one, because we process the course of the robot thanks to his previous position. Therefore, if the robot stops moving, the processed course is wrong and all the regulation is compromised. To ensure that, we maintain the forward command slightly over 1500. The turn command however needs to have the same degree of freedom below and beyond the neutral point of 1500. So we linearly transform (u_v,u_θ) into a couple of PWM values, of range [1600,1800]x[1000,2000]. We used a smoother transition than the linear transformation of a saturated command, which is the hyperbolic tangent. Finally, the command is saturated twice, the first time in "natural" unit that are understood by the simulator and the second time as PWM values that are used by the motors.

Chapitre 4

Implementation with buggy robots

4.1 Hardware

Maxime Bouyssou

4.1.1 Hardware architecture

The hardware part of each robot is pretty simple. Each robot is equipped with a serial GPS wired to a serial radio module (see figure below). So each robots will emits its GPS coordinates to the central computer who will then uses this coordinates to compute the next order he will send to them.

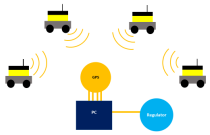


FIGURE 4.1 – Robot emitting their GPS coordinates

The radio modules are wired to the power electronic parts of the robot. The robot is controlled via a standard RC radio link. The idea is that the central computer can directly send his orders via the radio transmission to the robots. (see figure below)

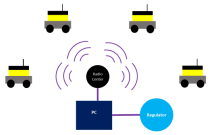


FIGURE 4.2 – Robot emitting their GPS coordinates

Here is a more details view of all the composants inside the robots.

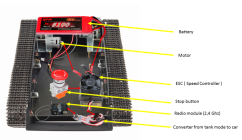


FIGURE 4.3 – Construction of the robot

Here is a more details view of all the composants on the ground station

FIGURE 4.4 – GPS

FIGURE 4.5 – Station Control

4.2 GPS Sentence Analysis

Pierre JACQUOT

In order to get the GPS coordinates of each robots we decided to use the *GPRMC* sentence send by the GPS's emitter. The exemple below show the structure of a *GPRMC* sentence :

UTC	Data status	Latitude	N or S	Longitude	E or W	Speed (knots)	Track made good	UT Date	Magnetic Variation	E or W and Checksum
081836	A	3751.65	S	14507.36	E	000.0	360.0	130998,01	011.3	E*62

The most valuable data are the latitude, the longitude, the speed over ground (in knots) ant the time stamp. In order to get those different data, we used the python package pynmea, which allows to easily get and parse a GPS sentence.
However, the longitude and latitude obtained using this specific sentence have a particular structure that we had to changed to make them easier to manipulate and compute in our algorithm. The example below show how the longitude and latitude are originally formatted :

— Longitude : *12311.12,W* which means Longitude 123 degree. 11.12 min West
— Latitude : *4916.45,N* which means Latitude 49 degree 16.45 min North

In order to ease the computation, our program automatically convert the longitude and latitude in degrees, take into account the cardinal direction associated with each coordinates. These cardinal directions North and South for the latitude and East and West for the longitude, let us know respectively on which side of the Greenwich (or prime) meridian we are located and on which side of the equator we are located. As a matter of fact, we will have in Brest a "negative" latitude and a positive longitude, due to our positioning regarding the equator and Greenwich meridian.

Our program not only give the coordinates in degrees but also convert them into UTM (Universal Transverse Mercator) coordinates. This new set of coordinates allow a localisation of the robot in a flat local coordinates system and maybe more easy to use as they are just decimal numbers. The UTM system itself, consist in a subdivision of the world in different sectors which can be consider as flat, and with a specific system of coordinates. The figure below show the subdivision of France :

FIGURE 4.6 – Secteur UTM

This figure show that we are currently in the sector 30. Whereas,in order to compute UTM coordinates, we need first to choose a geodesic system representing the Earth. For this project we chose the WGS-84 system which is the most commonly used system.

4.3 Structure of the programming code

Philippe CHUZEL

The code which have been implemented for this part of the project can be sum up with the following picture 4.7 :

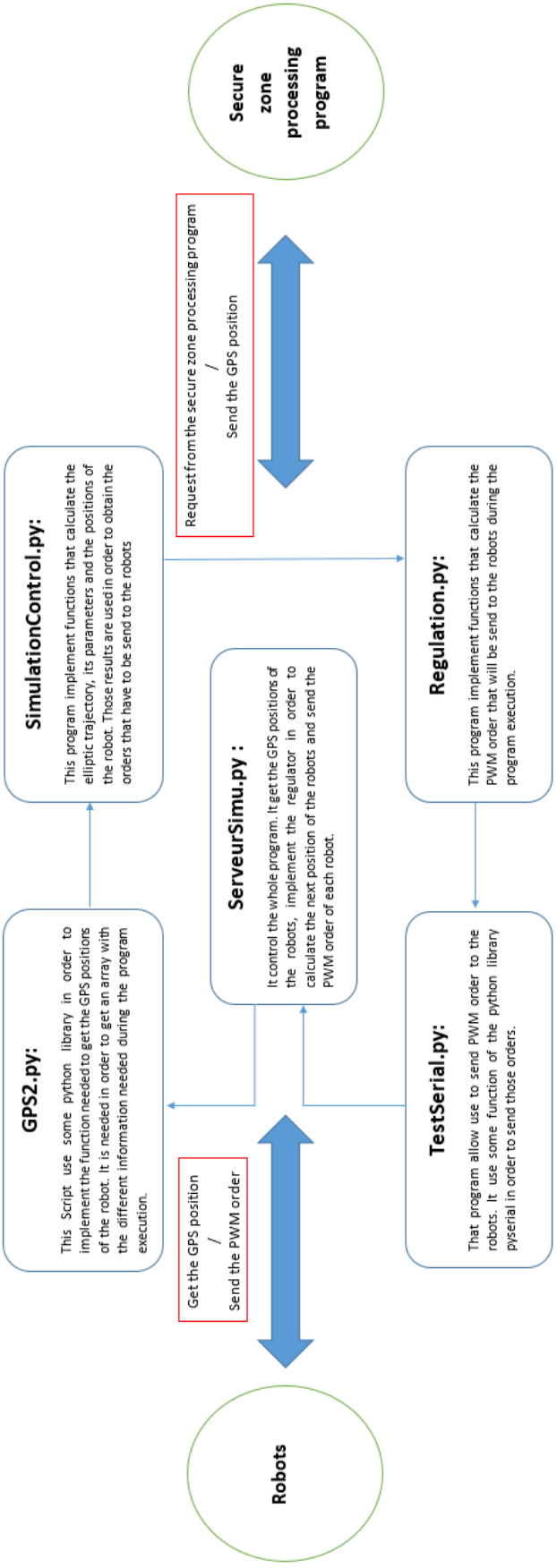


FIGURE 4.7 – Architecture of the program which control the regulation of the robots.

As you can see, there are 5 main script that are used in order to run the whole program.

Four are used in order to implement only a part of the functions needed :

— **GPS2.py** : this script has been implemented by *Pierre JACQUOT* and allow us to get access to the GPS signal of all the robots. This program catch the first available GPS tram of each robot, put it in a numpy array. The parameters which are kept are the latitude, the longitude, the speed, the north, the east and the time. The current version allows us to realize a multi-threading program in order to get every GPS positions. In order to call every function, we have to instantiate an object call ***"ThreadGPSAll"***

— SimulationControl.py has been implemented by *Alice DANCKAERS* and is used here in order to get the elliptic trajectory in function of the instant t and the theoretical position of each robots.In order to call every function, we have to instantiate an object call ***"SimulationControl"***.

— Regulation.py has been implemented by *Thomas BOULIER and Sylvain HUNAULT* and is used here in order to get PWM order which have to be send to every robots. Each robots has a PWM in order to go forward and another one in order to turn. It use the theoretical position and the current position of the robot and it return the PWM values. It implement the object ***"Robot"*** and thanks to it, we are able to keep all the information of the group of robots. Furthermore, it allows us to use the function which are needed in order to regulate every robot.

— testserial.py has been implemented by *Benoit Raymond* and is used here in order to send command to every robot. We are using a serial communication in order to send those order to a HF transmitter. Once again, , we have to instantiate an object call ***"command"*** in order to use the functions in this script.

The last one is here in order to organize the whole program and it's called Serveur-Simu.py. It use one thread in order to :

— Get the current GPS positions of every robots

— Get the new elliptic trajectory and give the instructions to every robots

— Get the new PWM command that has to be apply to every robots.

— Send those PWM instruction.

The other thread is here in order to allow a communication between the robots and the program which has to run the secure zone processing algorithm.

This script is the master and all the other script are just intermediary in order to regulate the robots.

The following picture sum up how this script organize the whole program.

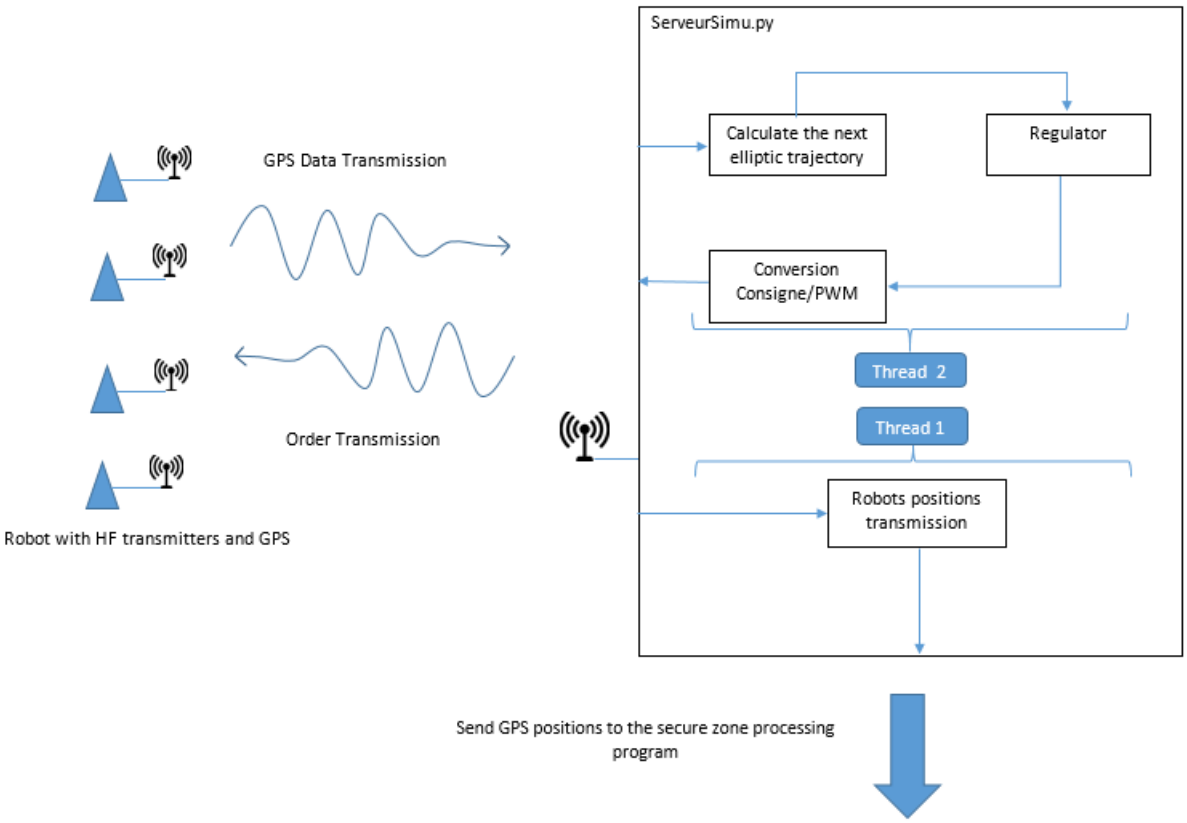


FIGURE 4.8 – Architecture mise en place pour la simulation sur robot char.

4.4 Results

Conclusion

Appendix

Annexe A

Appendix 1

Annexe B

Appendix 2

Annexe C

Appendix 3