STATUS REPORT

Surveillance Of The Bay Of Biscay

Written by :

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

Under the supervision of :

Luc Jaulin
Benoit Zerr

ENSTA Bretagne

Option Systèmes Perception Information Décision

Contents

Acknowledgment

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

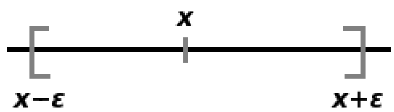
Secure zone processing

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 :

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

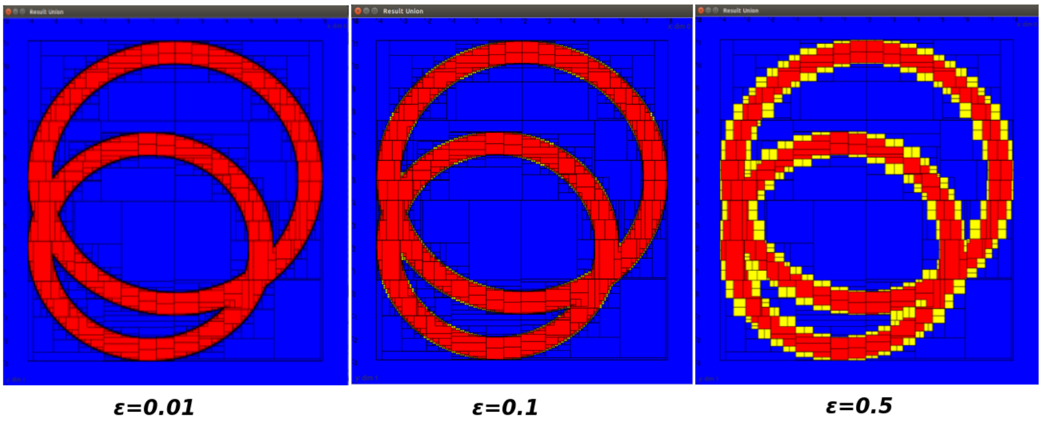


FIGURE 2.1 – Resolution of S1 U S2 for different precision with intervals calculation

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.

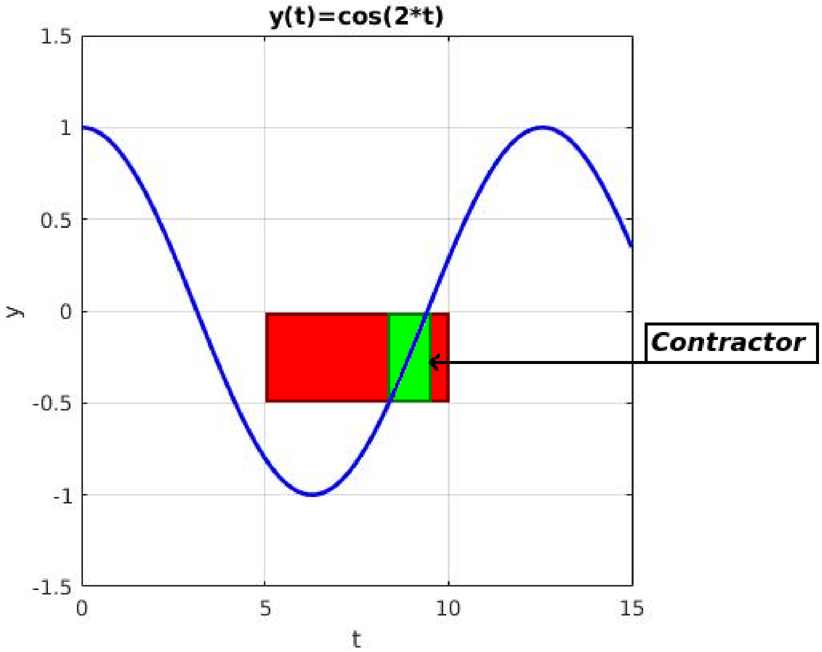


FIGURE 2.2 – Example of the utilisation of contractor

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

2.2.1 Basics and Usefulness of Interval Arithmetics

Normal solver equation are efficient with linear sytems but in a real environment

2.2.2 Thick Functions

Introduce the concept of MAYBE

2.2.3 Special Test for Gascogne Surveillance

We found a test to determine if a box is inside the secure zone covered by one

Algorithm 1 Is $\mathbf{X} \subseteq \mathbb{S}$, $\mathbb{S} =$ Secure Zone and $\mathbf{X} \in \mathbb{R}^2$

Require: X,$range$ (reach of boat),m (position of boat)

$Xmx \leftarrow max([0,0], sign((X[0]-m[0].ub())*(X[0]-m[0].lb())))* min((X[0]-m[0].lb())^2, (X[0]-m[0].ub())^2)$

$Xmy \leftarrow max([0,0], sign((X[1]-m[1].ub())*(X[1]-m[1].lb())))*min((X[1]-m[1].lb())^2, (X[1]-m[1].ub())^2)$

$Xm \leftarrow Xmx + Xmy$

$Xp \leftarrow max((X[0]-m[0].lb())^2, (X[0]-m[0].ub())^2) + max((X[1]-m[1].lb())^2, (X[1]-m[1].ub())^2)$

$Xub \leftarrow Xp|Xm$

if $Xub \cap [0, \text{range}^2] = \emptyset$ **then**
  return OUT
else if $Xub \subseteq [0, \text{range}^2]$ **then**
  return IN
else
  if $\text{range}^2 - Xp.ub() < 0]$ **then**
    if $Xm - \text{range}^2 \subseteq [-\infty, 0]$ **then**
      return MAYBE
    else
      return UNKNOWN2
    end if
  else
    return UNKNOWN
  end if
end if

Differente result In testing :

Symbol	Meaning
0	Box is not in the Secure Zone
1	Box is in the secure zone
?	Box may be in the secure zone
[0, ?]	Box may be in the secure zone
[?,1]	Box may be in the secure zone
[0,1]	Box may be in the secure zone or out

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]

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)
  $R \leftarrow$ Algorithme1(X,range,{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

2.3 Mesh to boxes functions

2.4 Erosion functions

2.5 Global Sivia algorithm

2.6 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 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. The second approach is a method based on artificial potential 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.

3.1.1 Artificial potential field

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 potential field between them. In case of obstacles, we can consider them as particles of same charge than the robot's. The potential field method calculate the instantaneous speed vector $w(p_{robot}, t)$ the robot need 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 :

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

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

A proportional regulation compute the command u_v and u_θ which will be used to control the robot.

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

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

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 use it to compute the new position of each one.

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

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 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

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. the command taken by the robots is a PWM values in the range [1000,2000]. Because we need to keep the robots always moving, we do not want to allow the robots to have a null speed or worst, a negative one. 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 [1600,2000]x[1000,2000]. We considered using a smoother transition than the linear transformation of a saturated command, such as the hyperbolic tangente, but we finally decided that it was not necessary.

Chapitre 4

Implementation with buggy robots

4.1 Matériel

Pour la réalisation d'une simulation sur des robots, nous avons du implémenter un architecture afin de permettre à nos programme de gérer d'une part nos robots et d'autre part de voir si le résultat théorique correspond aux résultats expérimentaux.

Nous aboutissons au final à l'architecture suivante :

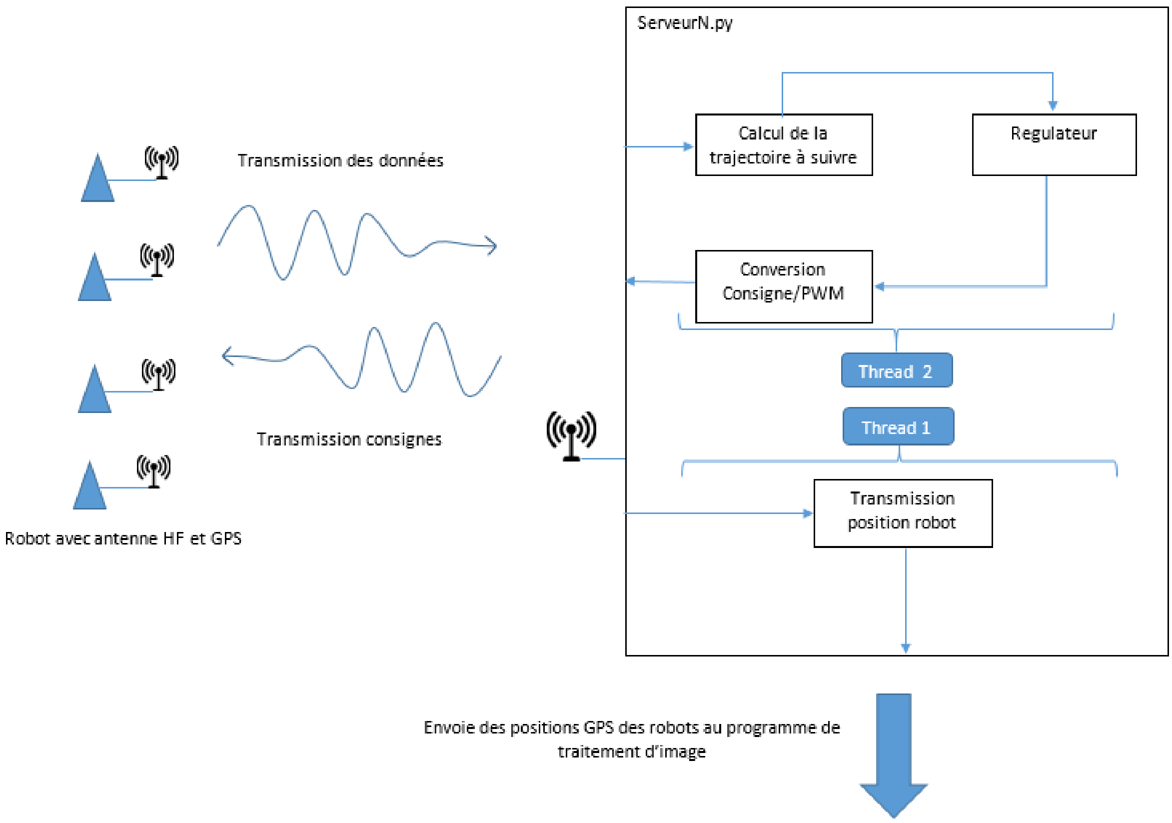


FIGURE 4.1 – Architecture mise en place pour la simulation sur robot char.

4.1.1 GPS

ici la description des GPS utilisés

4.1.2 Robot

ici la description des robots et des récepteurs

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 :

081836	A	3751.65	S	14507.36	E	000.0	360.0	130998,0	1011.3	E*62
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

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 4.2 – 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 most commonly used.

4.3 Structure of the programming code

Quatre scripts python : GPS2.py, commande.py, testserial.py et ServeurN.py

The code which have been implemented for this part of the project can be sum up with the following picture 4.3 :

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

— **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 programm catch the first available GPS tram og each robot, put it in a numpy array. The parameters which are keept are the latitude, the longitude, the speed, the north, the east and the time.

— 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.

4.4 Résults

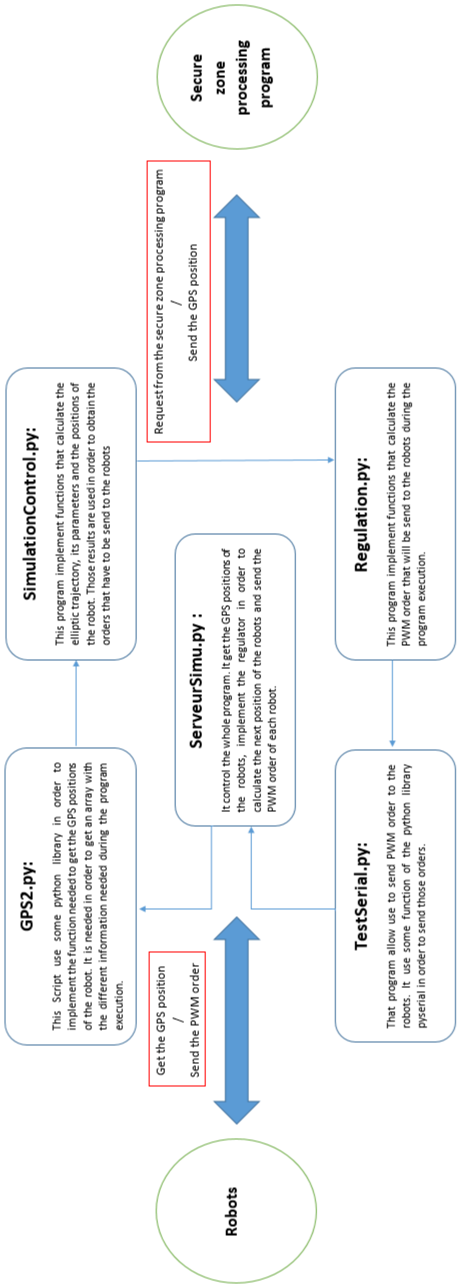


FIGURE 4.3 – Architecture of the program which control the regulation of the robots.

Conclusion

Appendix

Annexe A

Appendix 1

Annexe B

Appendix 2

Annexe C

Appendix 3