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

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 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 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

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

3.2 Regulation of the robot pack

Two methods have been studied and implemented to regulate the pack of robots. The first is a method by artificial potential field, robust and easy to debug. This method was chosen to regulate the real robots on the play field. The second method is a method of looping linearisation, very efficient. This method was chosen for the simulation because its integration in a simulation is easier than on robots.

3.2.1 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 a surcharge of the actionners.

3.3 Evolution of the elliptic trajectory

3.4 Conversion of the order to a PWM command

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 :

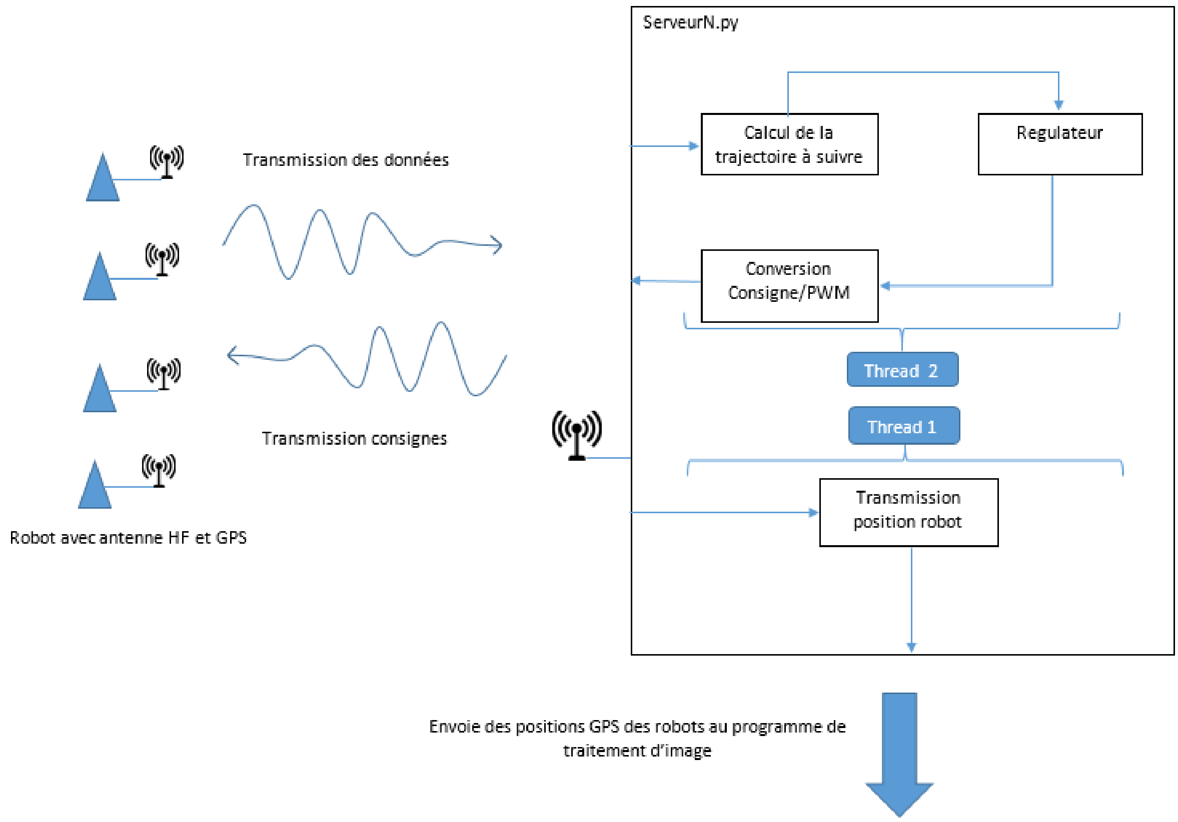

FIGURE 4.1 – Architecture mise en place pour la simulation sur robot char.

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,	011.3	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 du code

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

4.4 Résultats

Conclusion

Appendix

Annexe A

Appendix 1

Annexe B

Appendix 2

Annexe C

Appendix 3

Bibliography