Rapport d'avancement

Surveillance du Golfe de Gascogne

rédigé par :

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

sous la direction de :

Luc Jaulin
Benoit Zerr

Option Systèmes Perception Information Décision

Sommaire

Remerciements

Chapitre 1

Introduction

Le but de ce projet est de détecter des intrus dans une zones maritime donnée. Ici, nous allons nous intéressé au Golfe de Gascogne. Nous allons donc utiliser un groupe de robots équipés de GPS et nous allons implémenter des algorithmes et un stratégie pour assurer la sureté d'une zone.

Nous allons avoir plusieurs hypothèses de départ :
- Nous supposeront que l'étude peut être simulée dans un monde en deux dimensions.
- Les intrus auront une vitesse constante tout au long de la simulation.
- Tous les robots pourront voir autour d'eux sur une distance d_i et détecter si un intrus est présent sur cette zone.
- Nous définissons une zone sûre comme un ensemble de points où un intrus ne peut se trouver.

Nous allons donc présenter au final deux livrables :
- Une simulation théorique faite en grande partie en langage python qui permettra de déterminer la pertinence de nos algorithmes.
- Un simulation sur des robots chars disponibles au club robotique de l'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

Répartition des bateaux Dans un premier temps, nous avions penser à répartir les bateaux avec un espacement régulier selon l'abscisse curviligne de l'ellipse. Cette méthode étant finalement très difficile à mettre en place d'un point de vue mathématique et algorithmique, cette solution a été abandonnée. Il aurait fallu, pour se faire, approximer les intégrales de Wallis par un développement limité assez complexe, ce qui implique un temps de programmation important et un temps de calcul beaucoup trop long pour une simple étude de faisabilité. C'est pour cela que nous avons décidé de nous orienter vers un retard angulaire sur chacun des robots. Ainsi, chaque robot suit le robot précédent avec un retard angulaire fixe dépendant du nombre de robots parcourant l'ellipse. Cela implique que les robots vont être d'autant plus proches les uns des uns qu'il sont près des apogées de l'ellipse, et d'autant plus éloignés aux périgées. Néanmoins, malgré la non-linéarité du placement des robots, nous n'avons pas de rupture de la zone de détection. En effet, comme les robots vont plus vite aux périgées, la trace qu'ils "laissent" est plus grande, et inversement aux apogées. De ce fait, on peut garantir la sécurité de la zone à surveiller, même si les robots ne sont pas régulièrement espacés.

Chapitre 3

Robots regulation

3.1 Regulation

3.1.1 Regulation with potential field

3.1.2 Regulation with the linearisation of the command

This part has to be modified

3.2 Evolution of the elliptic trajectory

3.3 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 :

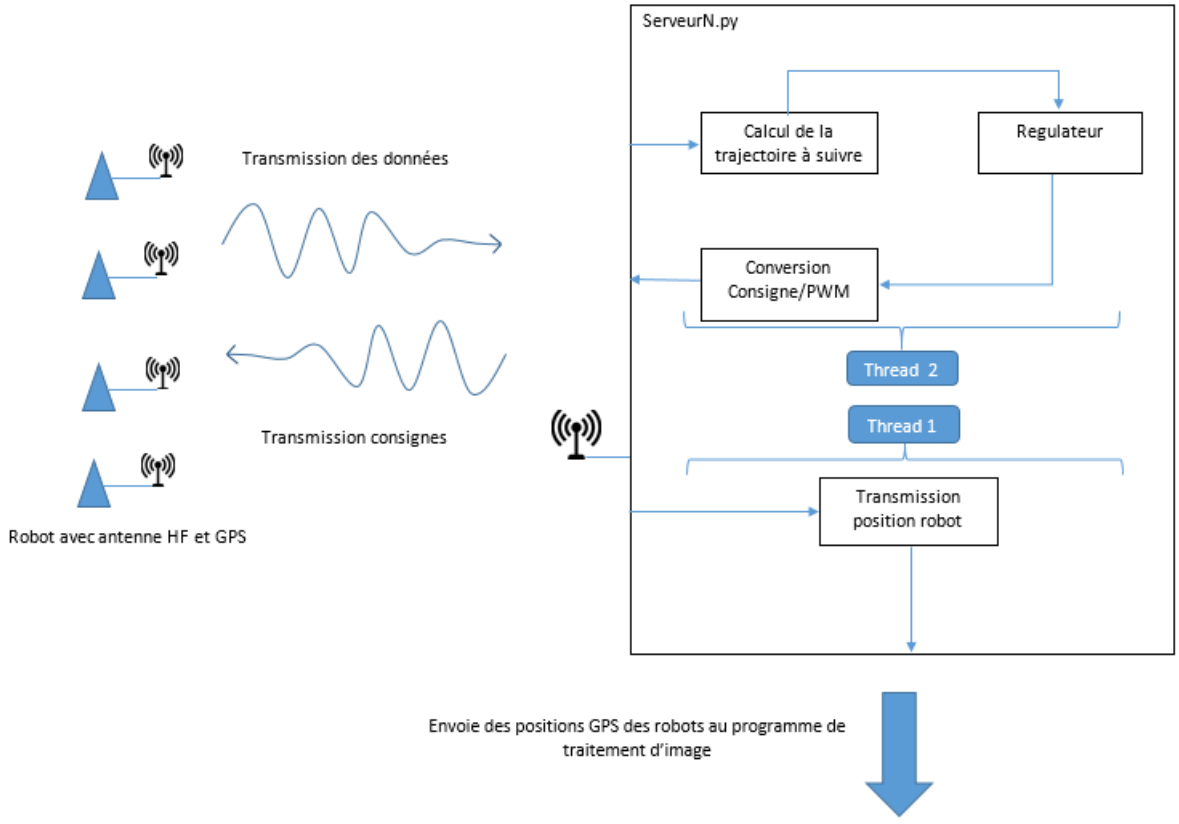

FIGURE 4.1 – Architecture mise en place pour la simulation sur robot char.

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	130998,01	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 :

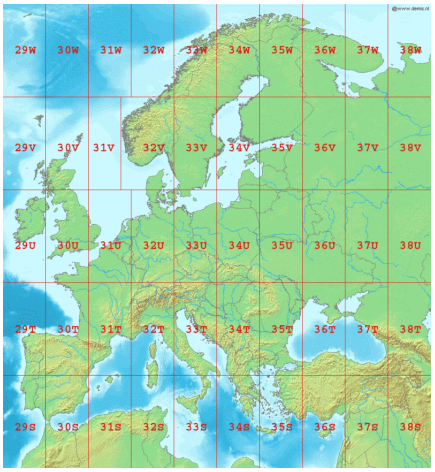

FIGURE 4.2 – Secteur UTM

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

Chapitre 5

Conclusion

Première partie

Annexes

Annexe A

Première annexe

Annexe B

Deuxième annexe

Annexe C

Troisième annexe