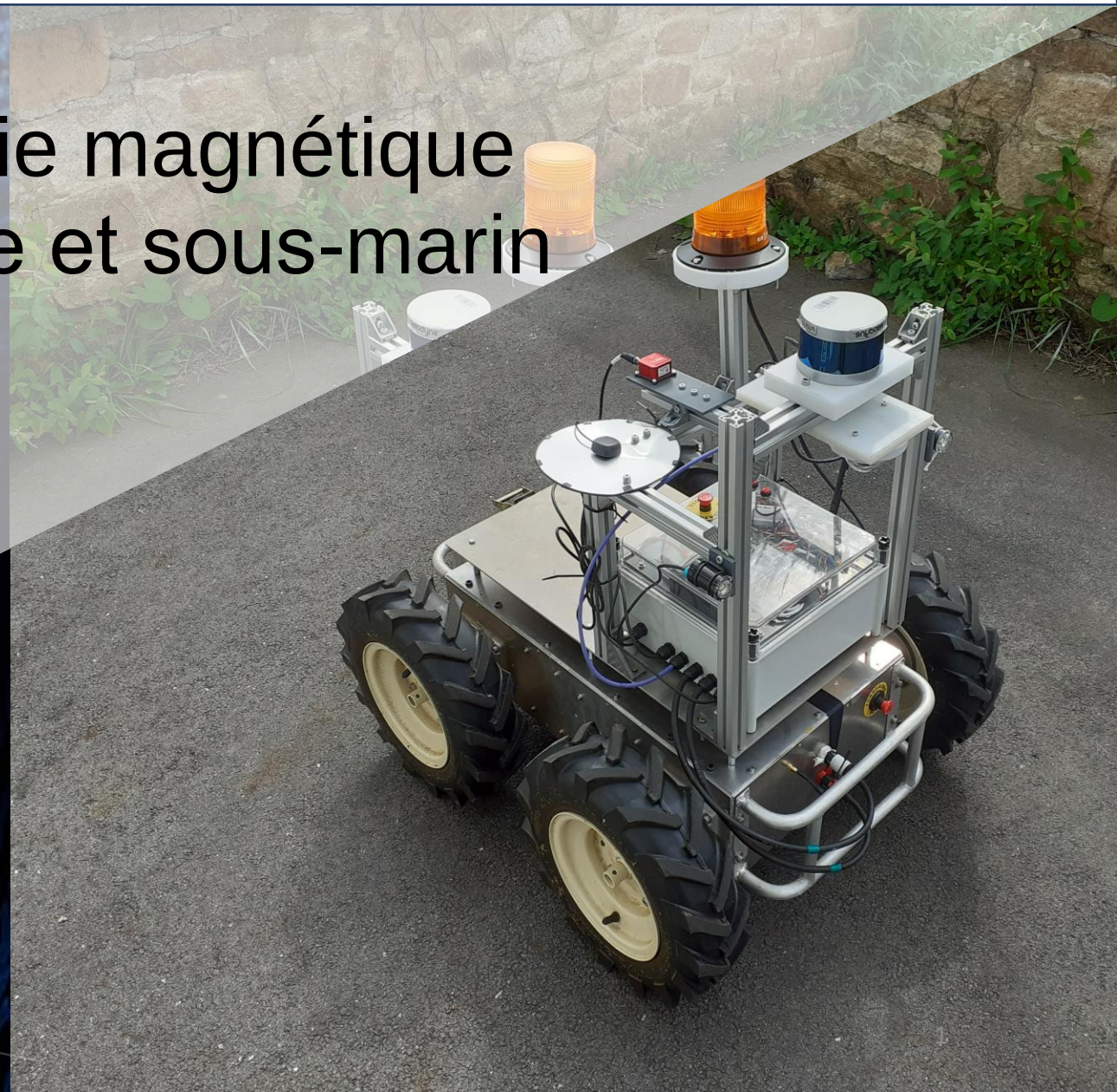
Projet Magmap : cartographie magnétique d'un environnement terrestre et sous-marin

U.E 5.4 : Architecture Robotique

Encadrant : M. Luc JAULIN



SOMMAIRE

Introduction

➢ **Objectif :**

Construire de manière **autonome** une **carte magnétique**.

➢ **Deux environnements :**

Sous-marin

Terrestre

19 Élèves-ingénieurs en voie d'approfondissement robotique autonome.

➢ **Intérêts :**

- Détection d'objets cachés ;
- Recherche d'épaves (Cordelière, …) ;
- Guerre des mines ;
- Etc.

Le magnétomètre

Magnétomètre : théorie

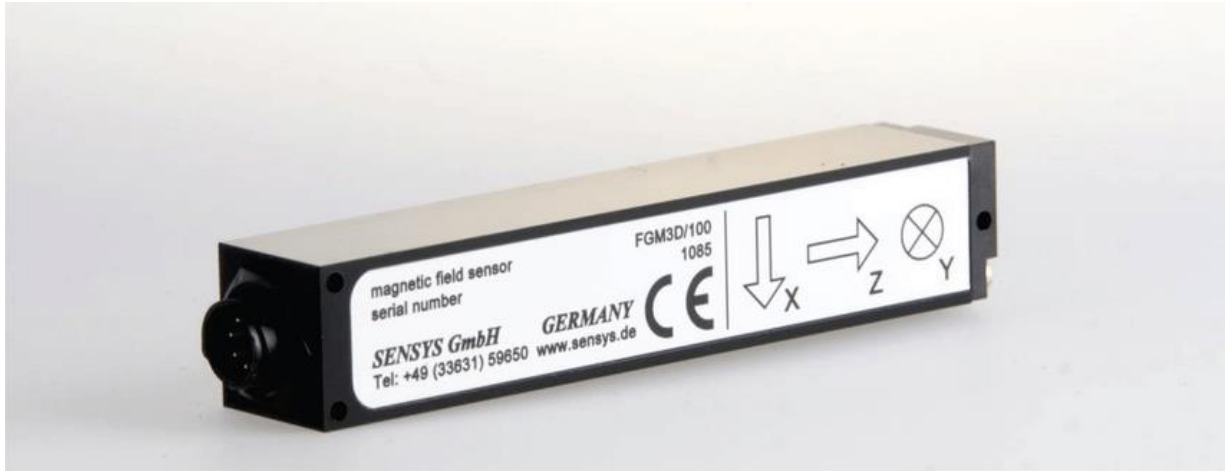
Présentation du magnétomètre utilisé

- Capteur de champ magnétique (Tesla)
  → aimantation
  → valeur du champ magnétique

- Magnétomètre : champ < 1 mT

- Tri-axial : mesure selon les trois axes

- Technologie fluxgate :
  → Grande sensibilité (~ 10 pT)
  → Faible consommation (3W)
  → Petite taille
  → Plage de mesure limitée (± 150 µT)
  → Bande-passante limitée



Le Sensys FGM3D 100 utilisé pour MagMap

Magnétomètre : théorie

Principe physique du magnétomètre fluxgate

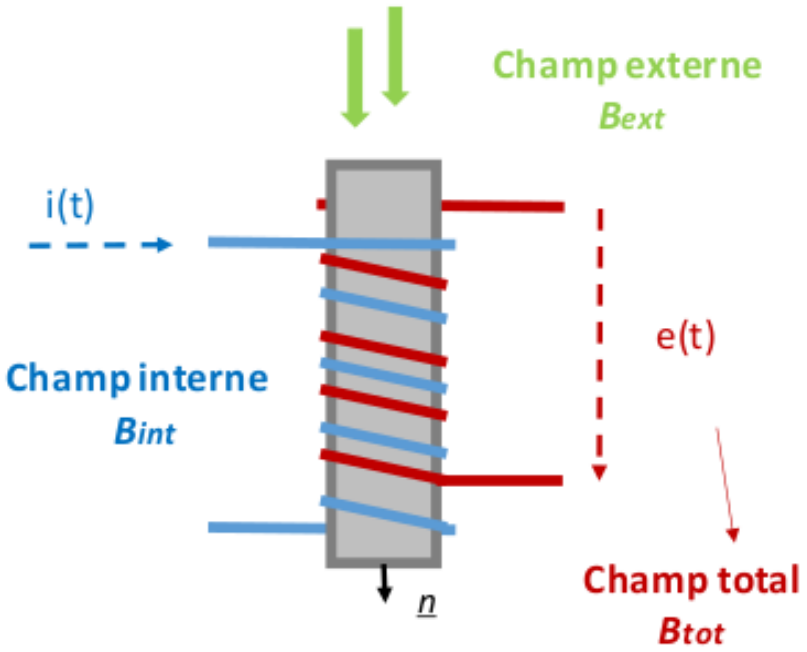


Schéma Fluxgate pour un axe

Loi de Faraday : $e(t) = -\frac{\mathrm{d}\phi(t)}{\mathrm{d}t}$

Φ : flux magnétique
e : force électromotrice induite

Or : $\phi(t) = \iint_s \overrightarrow{B_{tot}}(t) \cdot \vec{n}\, dS = nSB_{tot\perp}(t)$

n : nombre de spires
S : surface d'une spire

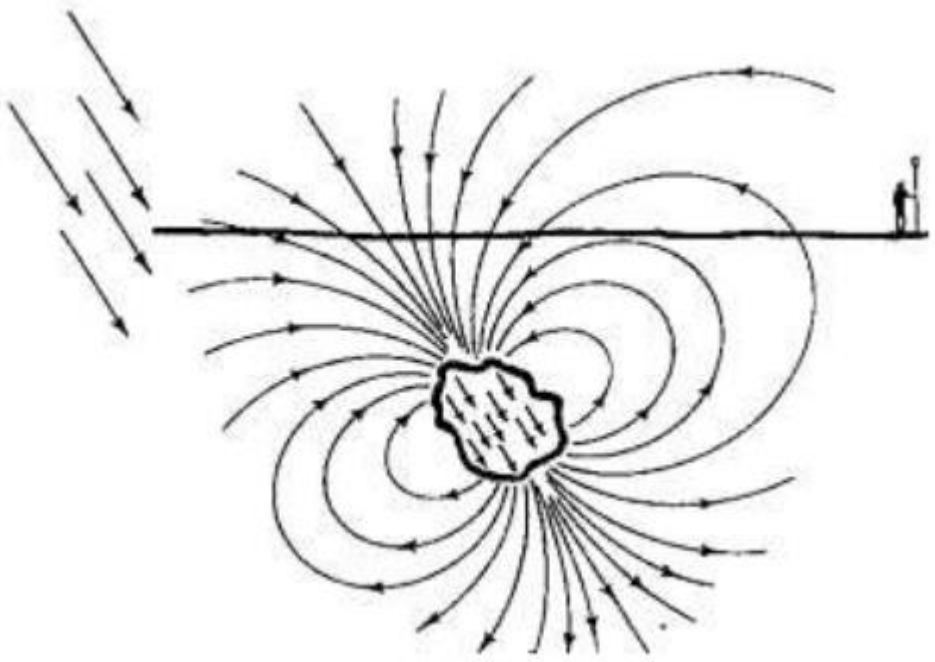
D'où : $e(t) = -\frac{\mathrm{dnSB_{tot\perp}(t)}}{\mathrm{d}t} = -nS\frac{\mathrm{d}B_{tot\perp}(t)}{\mathrm{d}t}$

$\leftrightarrow \quad B_{tot\perp} = \frac{1}{nS}\int_{t0}^{tf} e(t)dt$

$\leftrightarrow \quad \boxed{B_{ext\perp} = \frac{1}{nS}\int_{t0}^{tf} e(t)dt - B_{int\perp}}$

Utilisation d'un magnétomètre

Détection d'objets sous-terrain



Champ Magnétique Emis par un objet

Aimantation d'un objet:

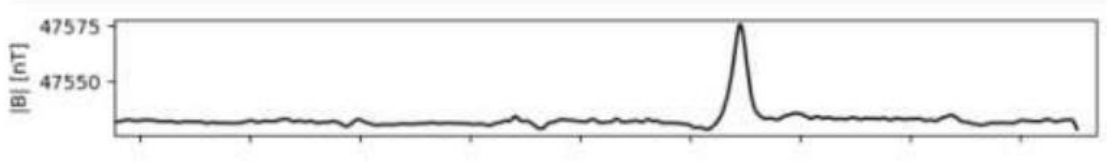
$$\vec{M} = \vec{M}_{permanent} + K_{eff} * \vec{H}_{Terre}$$

Induction magnétique perçue en un point:

$$\vec{B} = \frac{\mu_0}{4\pi}\left[3\frac{\vec{M}\cdot\vec{r}}{r^5}\vec{r} - \frac{\vec{M}}{r^3}\right]$$

Utilisation d'un magnétomètre

Détection d'objets sous-terrain



Anomalie détectée par le magnétomètre

Vérification empirique de la détectabilité:

$$\left|\vec{M}\right| = K \times Poids$$

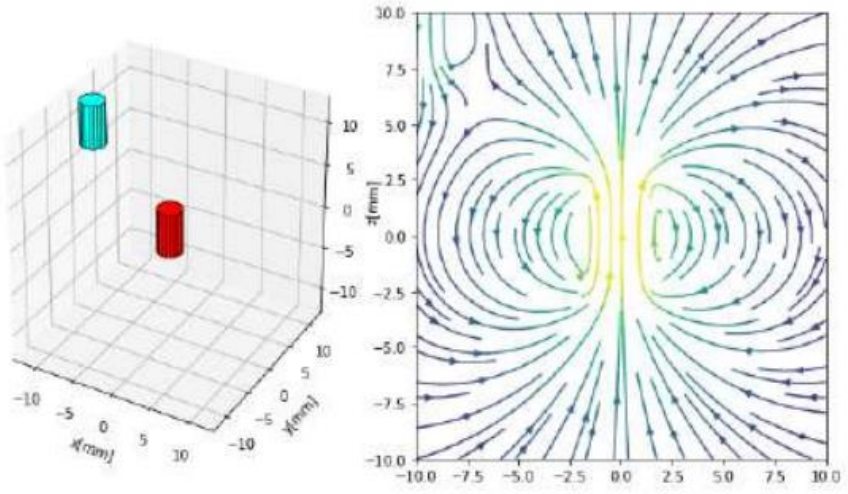
Avec K = 0.1 Am²/kg

$$\left|\vec{B}\right| = 100\frac{M}{r3}$$

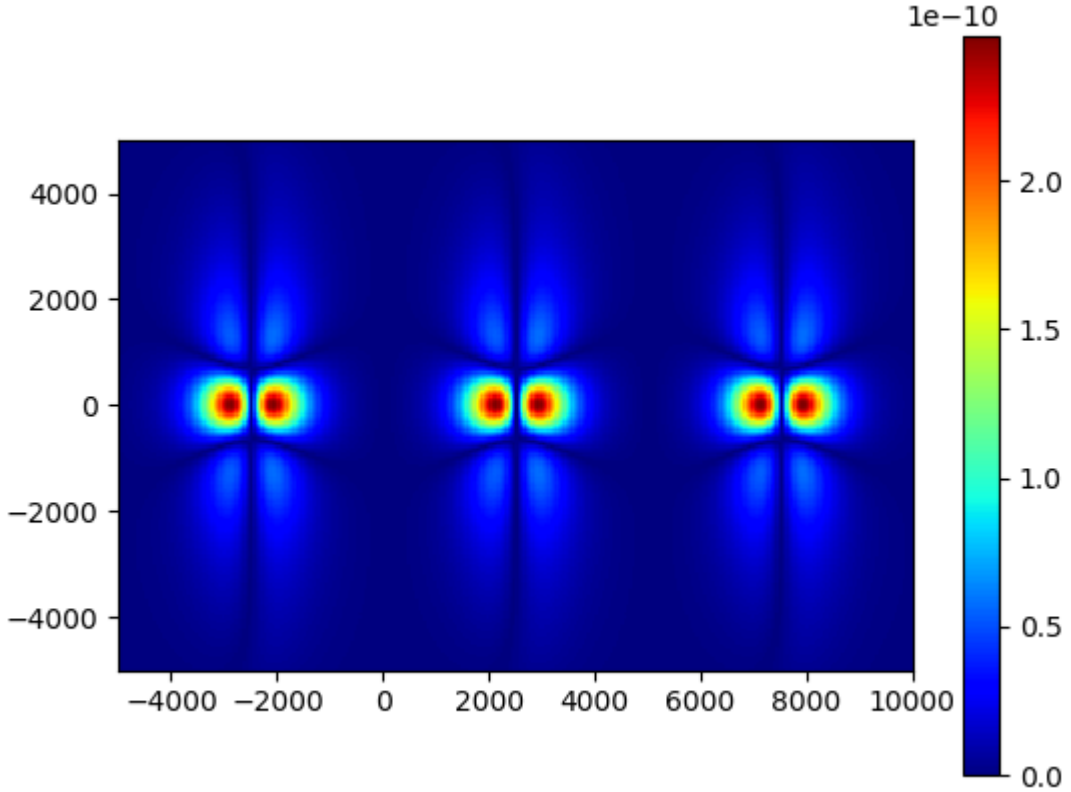
Pour un objet en acier de 100g à 30 cm de profondeur : on obtient comme valeurs:
M = 0.01 Am² et B = 37nT
->Théoriquement, l'objet est détectable par notre magnétomètre

Utilisation d'un magnétomètre

Simulation du magnétomètre



Configuration de la simulation

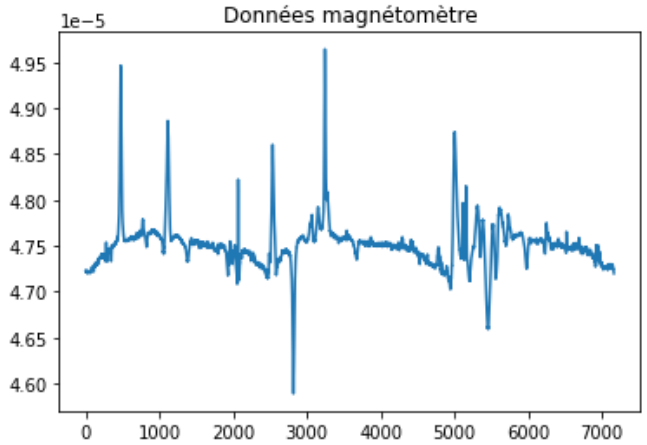


Détection de trois cylindres

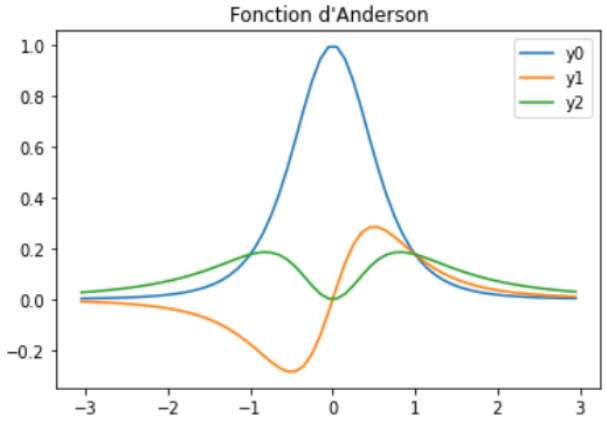
Magnétomètre : théorie

Filtre avec les fonctions d'Anderson

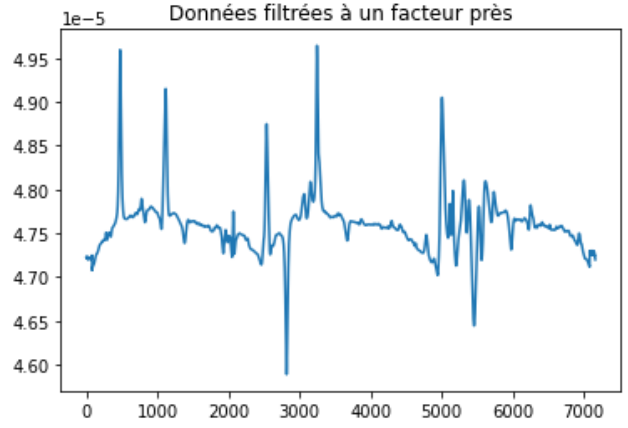
- Fonctions d'Anderson : $y_0(x) = \frac{1}{(1+x^2)^{5/2}}$ $y_1(x) = \frac{x}{(1+x^2)^{5/2}}$ $y_2(x) = \frac{x^2}{(1+x^2)^{5/2}}$
- Filtre d'Anderson :



Données brutes du magnétomètre



Corrélation croisée avec les fonctions d'Anderson



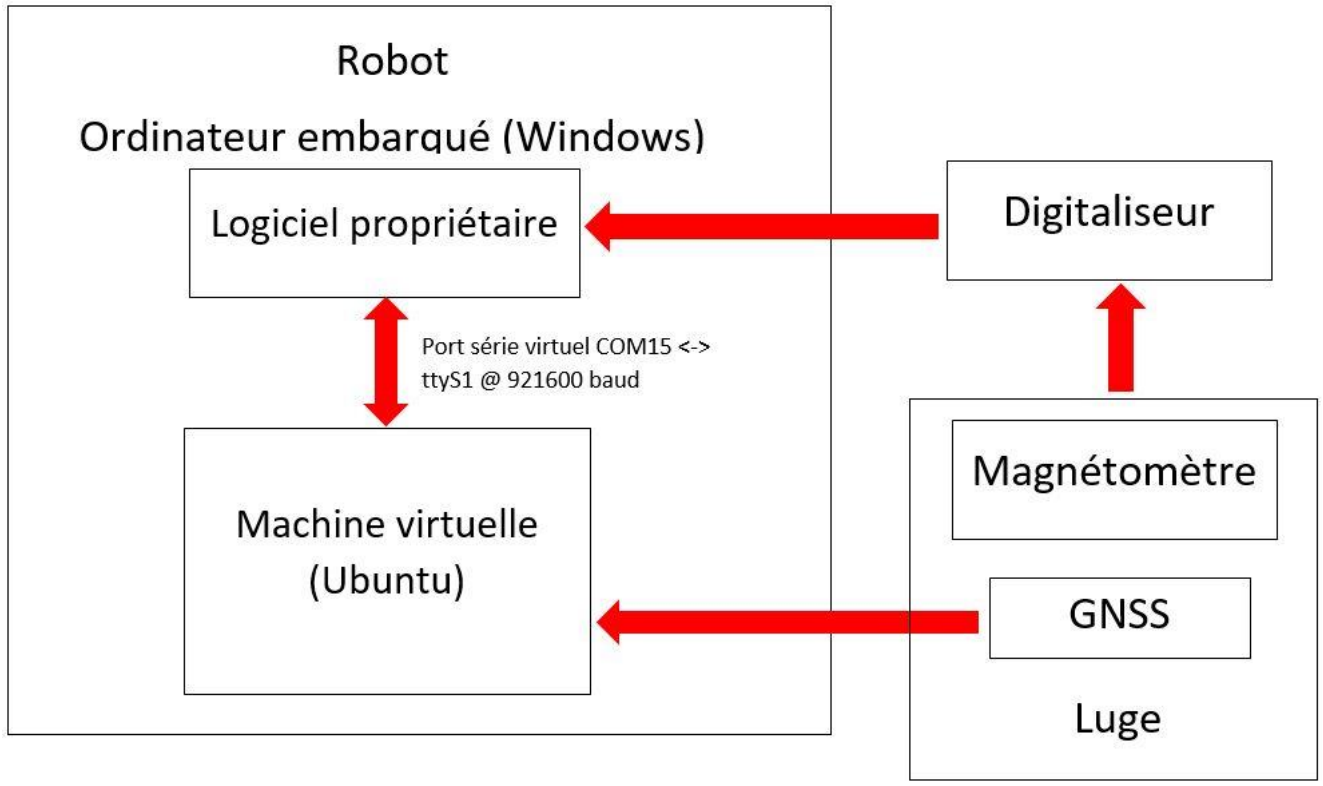
SIgnal reconstruit à partir des données théoriques

- Hypothèses non vérifiées en pratique ….

Magnétomètre : pratique

Connexion et récupération des données

- Utilisation du logiciel propriétaire FGM3D TD Application
- Synchronisation avec un récepteur GNSS



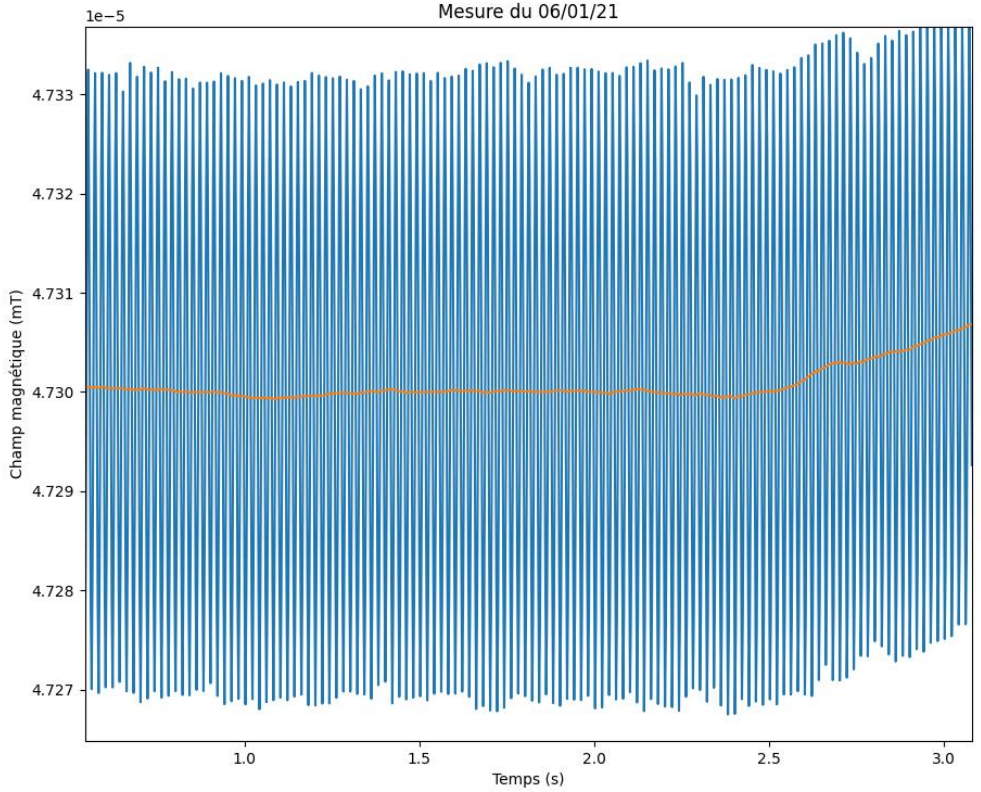
Magnétomètre : pratique

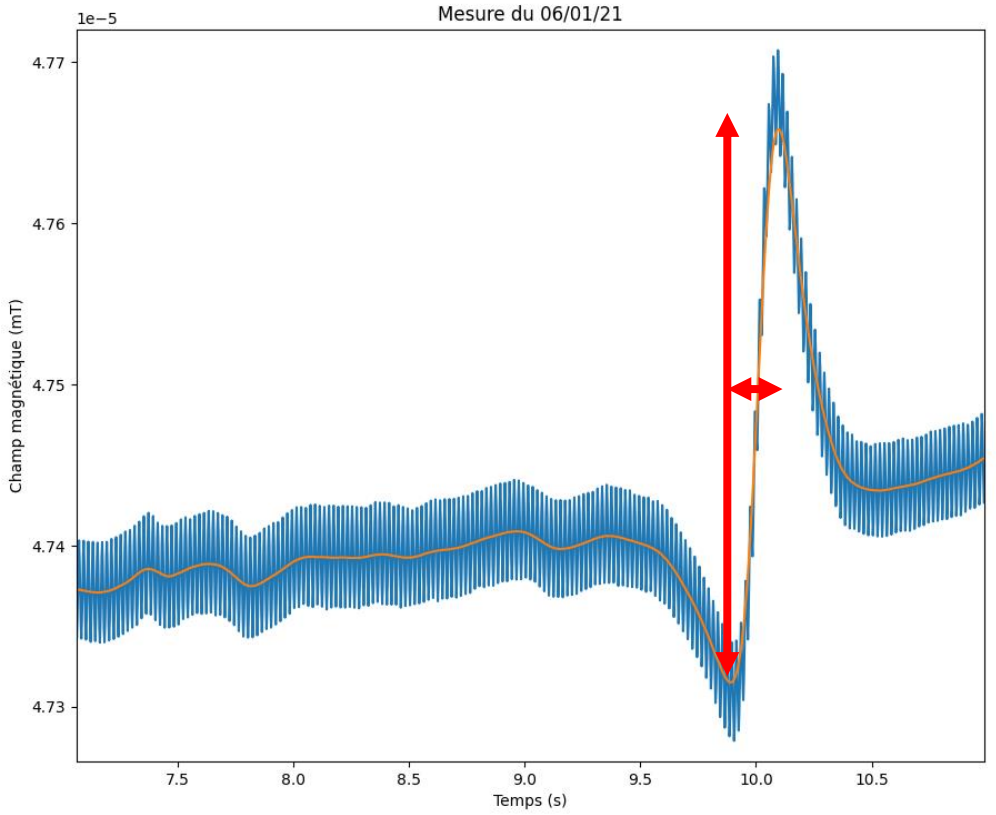
Montage sur le robot Saturne

- Montage du magnétomètre sur une luge
- Importance de la longueur de la corde
- Usage d'un poids

Magnétomètre : pratique

Traitement des données

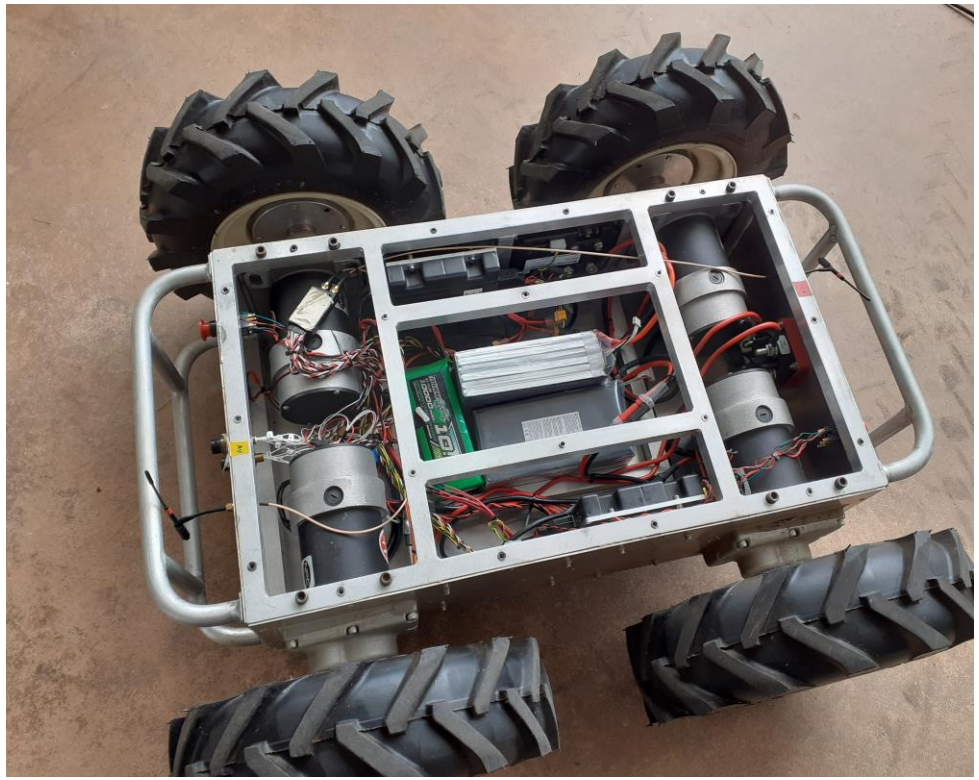




Prise en main des robots : Saturne

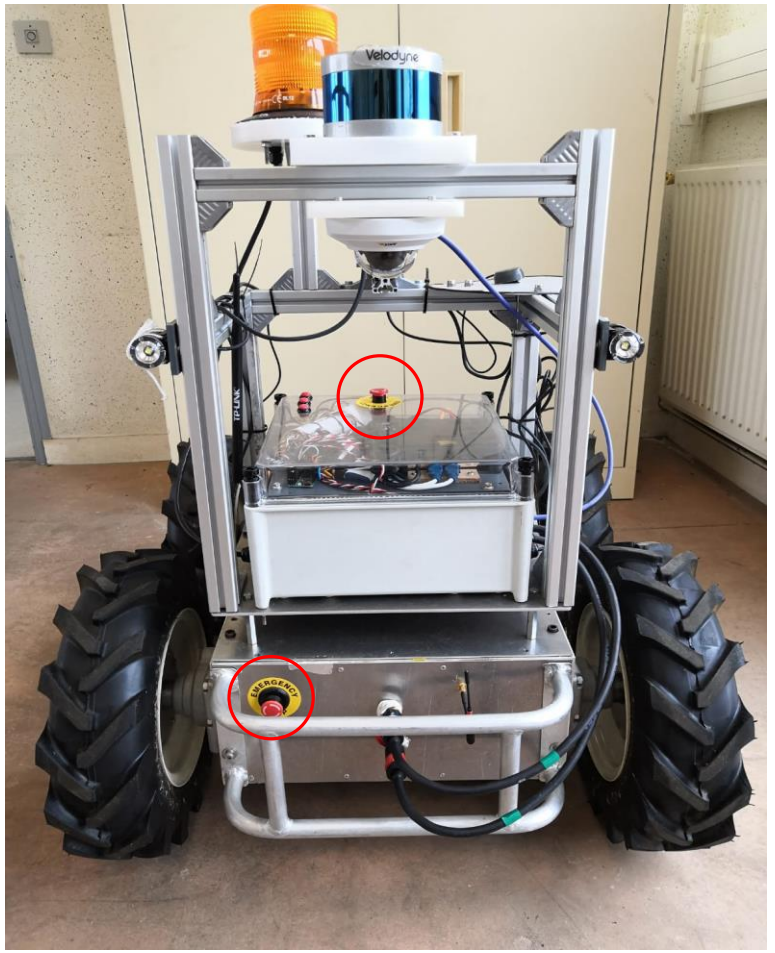
Présentation de Saturne :

- Fabriqué en grande partie à l'ENSTA Bretagne
- Composé de 2 parties indépendantes
- Etanche à la pluie (en principe)



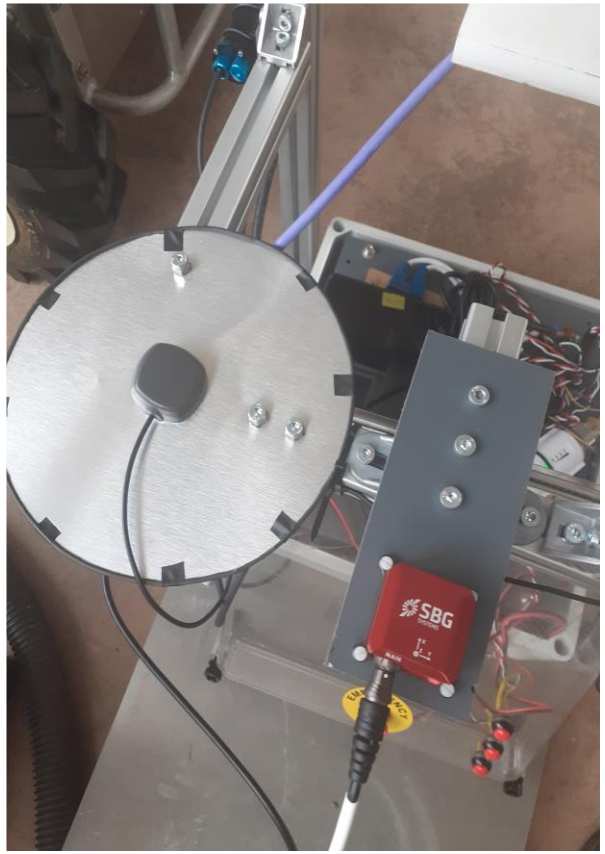
Moteurs, batteries et sécurité :

- 4 moteurs indépendants (500W chacun)
- 3 Batteries LiPo 6S (3 heures de mission)
- Trois arrêts d’urgence
- La télécommande prioritaire



Capteurs et électronique embarquée :

- GNSS compatible RTK
- Cartes de commande : mesure de tension et d'intensité
- Intel Nuc (S.E Ubuntu 18.04)
- LiDAR Velodyne Puck Lite
- Caméra IP



Prise en main de Saturne

Architecture logicielle

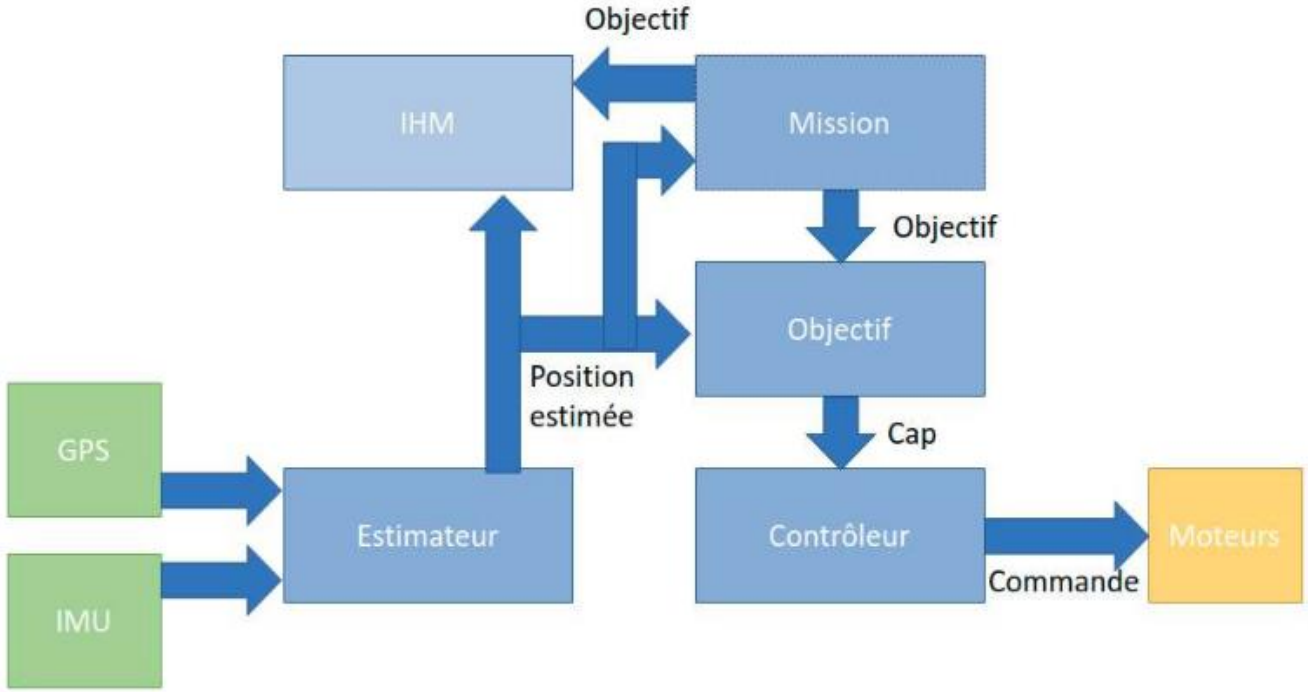
- Middleware → communication entre programmes informatiques
- ROS utilisé dans ce projet
- ROS = ensemble d'outils permettant le développement de logiciels pour la robotique

- Avantages ROS
  - Faciliter le travail collaboratif
  - Outil de visualisation, de simulation et de rejeu (rviz, gazebo ...)
  - Dispose d'une grande communauté

Prise en main de Saturne

Architecture logicielle

- Architecture ROS
- Niveau Hardware
  - Lecture GPS et IMU
  - Contrôle bas niveau
- Niveau observateur
  - Fusion des données
  - Estimation de la position
- Niveau commande
  - Pour une mission élementaire
  - Calcule du cap à suivre
- Niveau de la supervision
  - A partir d'un fichier texte
  - Gère les missions élementaires
- IHM



Prise en main des robots : Riptide

Riptide

Prise en main

- Robot de Riptide Autonomous Solutions (racheté par BAE Systems en 2019) ;
- Modèle utilisé dans le projet : **MKII microUUV** ;
- Trois parties :
  - La tête **: capteurs de navigation ;**
  - Le payload : **alimentation** + **CPU** ;
  - La queue : **actionneurs** + **antenne**.

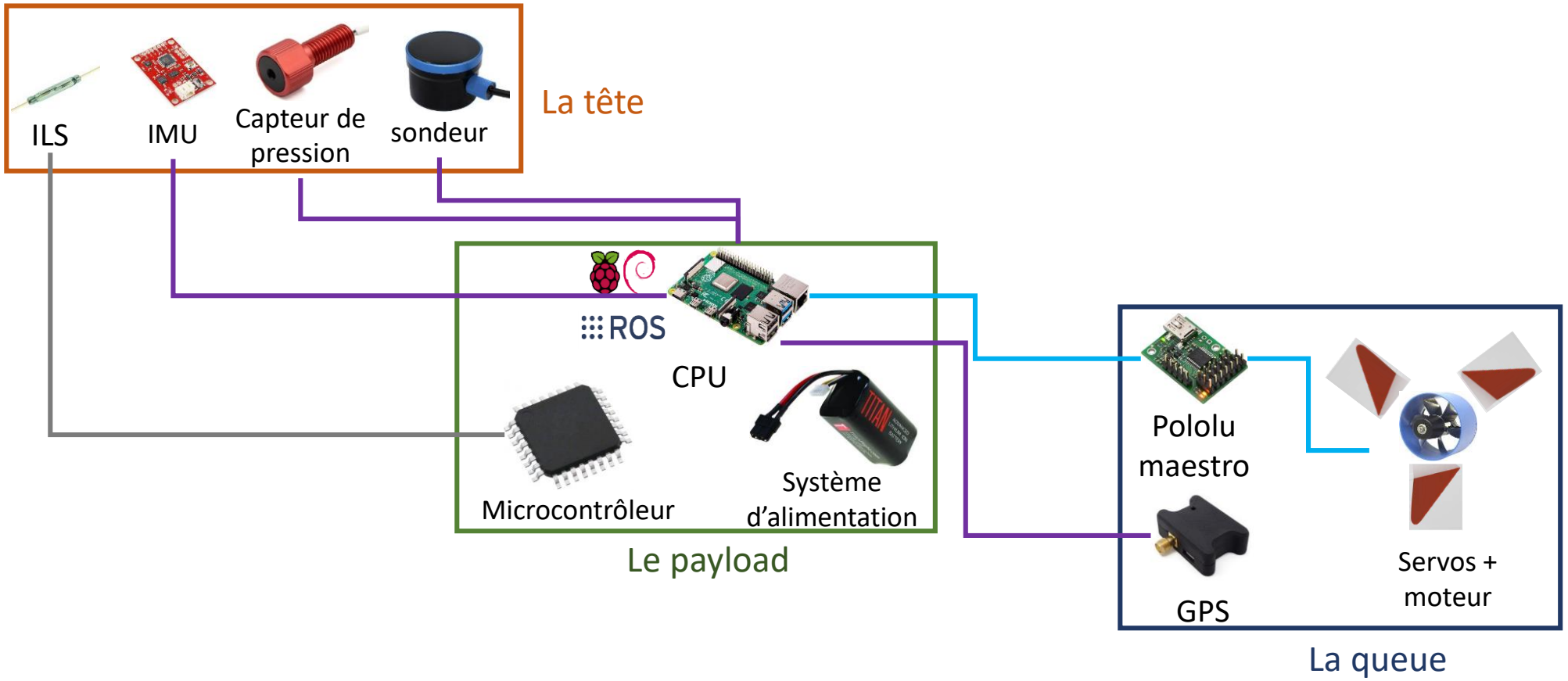
➢ **Inconvénients :** architecture et protocoles de communication propriétaire.

➢ **Travail effectué :**

- Nouvelle architecture matérielle ;
- Intégration nouveaux capteurs (GPS + capteur de pression) ;
- Système d'interruption magnétique ;
- Développement des drivers.

Riptide

Architecture matérielle



Riptide

Architecture matérielle

- **Système d'interruption magnétique :**
  - ILS (interrupteur à lames souples) et microcontrôleur PIC contrôlant un MOSFET ;

ILS

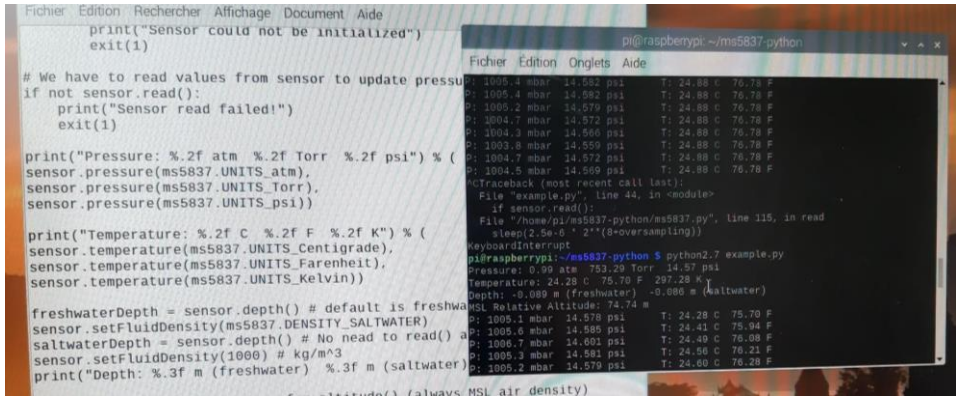
Microcontrôleur

- **Télé-opération à distance :**
  - Multiplexage PWM et mixage des voies pour contrôler le robot à la télécommande ;
  - Arduino pour décoder le signal PPM et le mixer

Riptide

Architecture matérielle

- **Capteur de pression :**
  - BlueRobotics bar30 : profondeur maximale 300m, résolution 2mm ;



- **GPS :**
  - TinyRTK avec utilisation de l'antenne intégrée à l'aileron ;

Riptide

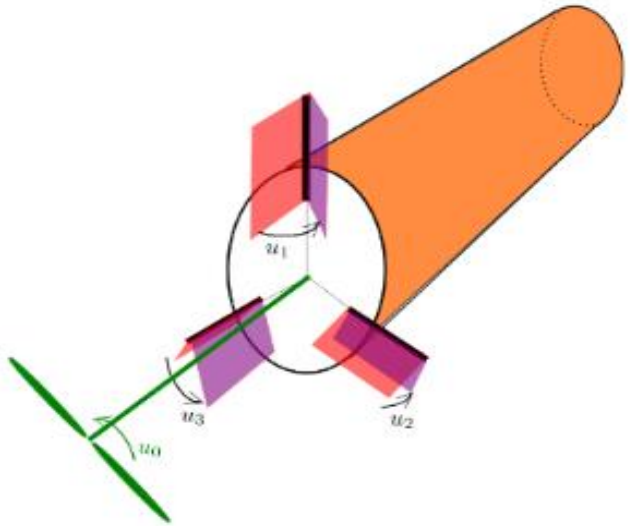
Architecture matérielle

- **Préparation aux missions :**
  - Configuration ssh ;
  - Configuration wifi ;
  - Installation et adaptation des drivers ;
  - Test des systèmes en simultané.

Riptide : contrôle

Equations

$$\begin{cases} \dot{p} = R(\varphi, \theta, \psi).v_r \\ \dot{v_r} = ar - wr\hat{}vr \\ \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & tan\,\theta\,sin\,\varphi & tan\,\theta\,cos\,\varphi \\ 0 & cos\,\varphi & -sin\,\varphi \\ 0 & \frac{sin\,\varphi}{cos\,\theta} & \frac{sin\,\varphi}{cos\,\theta} \end{pmatrix}.w_r \end{cases}$$



Riptide : contrôle

Equations

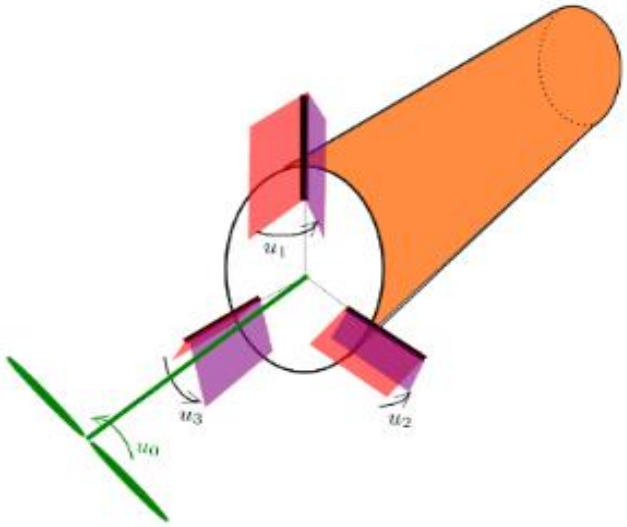
$$F_{pa} = \gamma \,.\, \upsilon \,.\, \sin u \,.\, a\vec{x}$$

- *F* la force exercée sur l'ailerons
- *u* l'angle d'inclinaison de l'aileron

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_x \end{bmatrix} = \upsilon . \mathrm{B} . \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

avec $B =$

$$\begin{pmatrix} p_3 & 0 & 0 \\ 0 & p_4 & 0 \\ 0 & 0 & p_4 \end{pmatrix} . \begin{pmatrix} -1 & -1 & -1 \\ 0 & \sin(2\pi/3) & -\sin(2\pi/3) \\ 1 & \cos(2\pi/3) & -\cos(2\pi/3) \end{pmatrix}$$



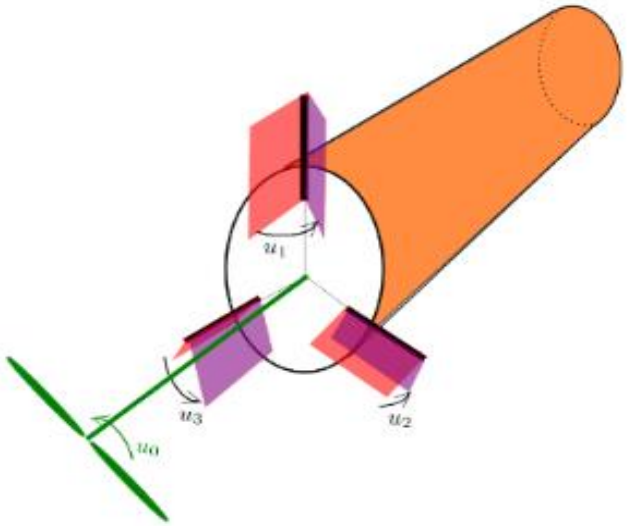
Riptide : contrôle

Equations

$$(m + m_a) \,.\, a = \alpha \,.\, u_0 \,.\, |u_0| - \beta \,.\, v_x \,.\, |v_x|$$

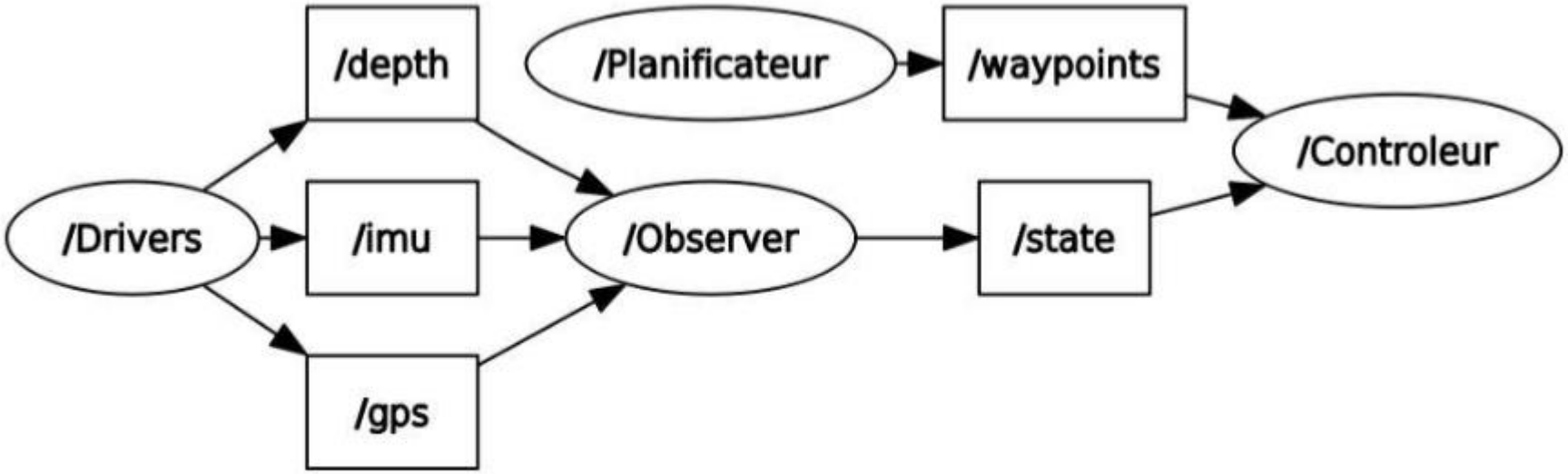
$$\Leftrightarrow a \;\; = \;\; p_1 \,.\, u_0 \,.\, |u_0| - p_2 \,.\, v_x \,.\, |v_x|$$

$$\begin{cases} \dot{p} = R(\varphi, \theta, \psi).\,(v_r, 0,0)^T \\ \dot{v_x} \;\; = \;\; p_1 \,.\, u_0 \,.\, |u_0| \;\; - \;\; p_2 \,.\, v_x \,.\, |v_x| \\ \begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & tan\,\theta\, sin\,\varphi & tan\,\theta\, cos\,\varphi \\ 0 & cos\,\varphi & -\, sin\,\varphi \\ 0 & \dfrac{sin\,\varphi}{cos\,\theta} & \dfrac{sin\,\varphi}{cos\,\theta} \end{pmatrix} . v_x \,.\, B(p_3, p_4). \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \end{cases}$$



Riptide : contrôle

ROS



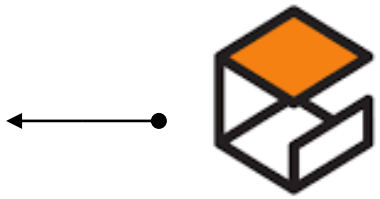
Simulation des robots

Simulation

Présentation de Gazebo/ROS

Simulateur 3D : Cinématique & Dynamique

Utilisé pour des compétitions technologiques et open-source : sous licence Apache 2.0

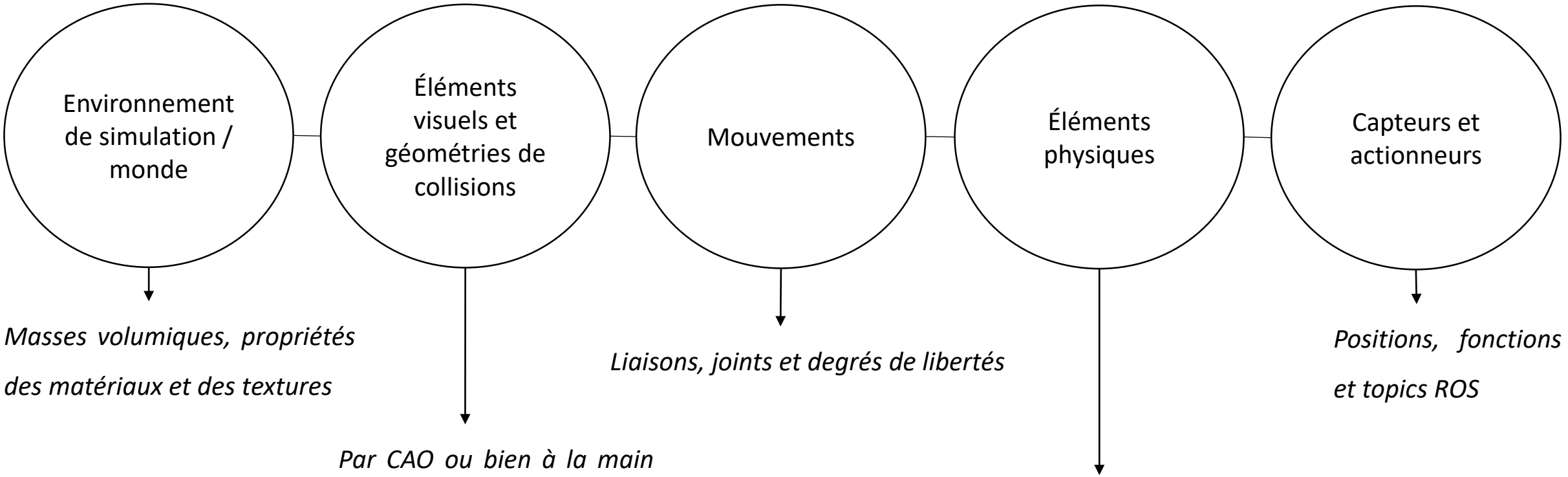


Interactions réalistes avec le monde

Compatible avec ROS : interfaçage avec capteurs et actionneurs

Simulation

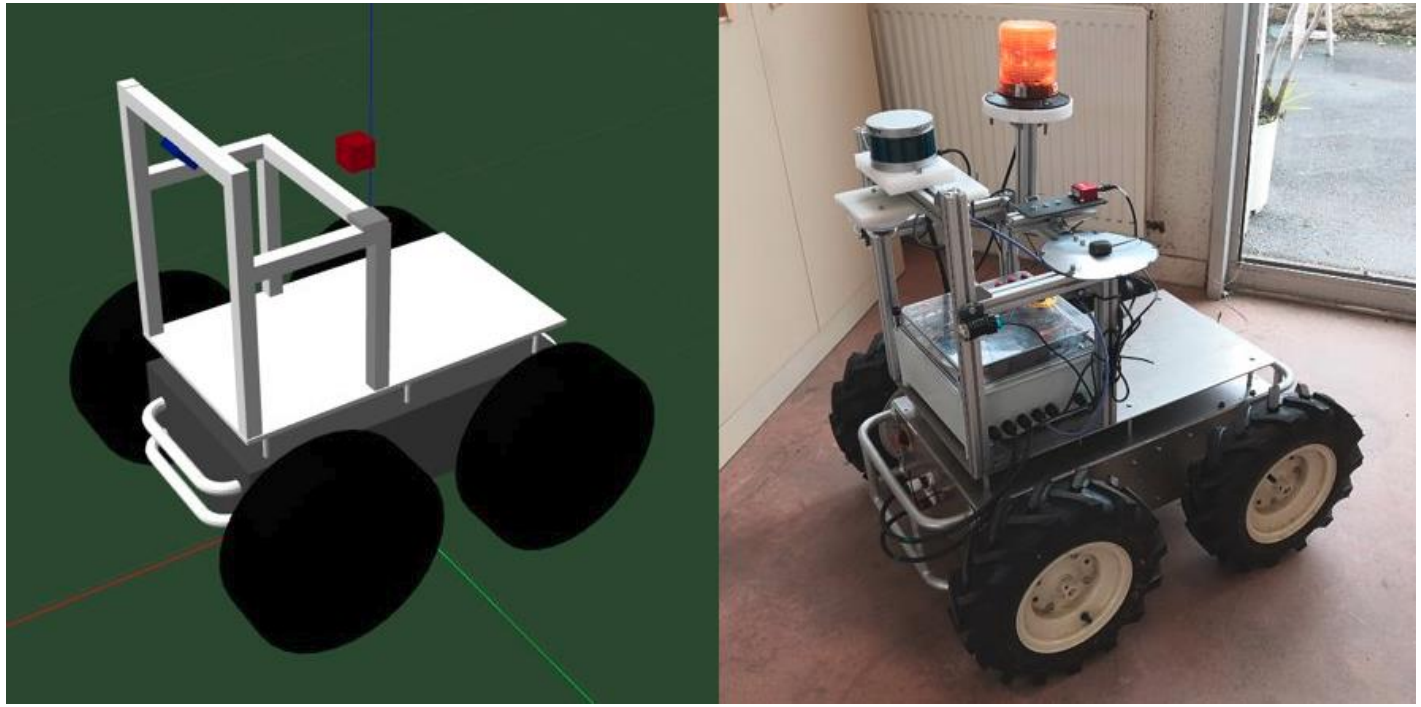
Les étapes pour simuler sous Gazebo



Simulation Saturne

Modélisation du robot

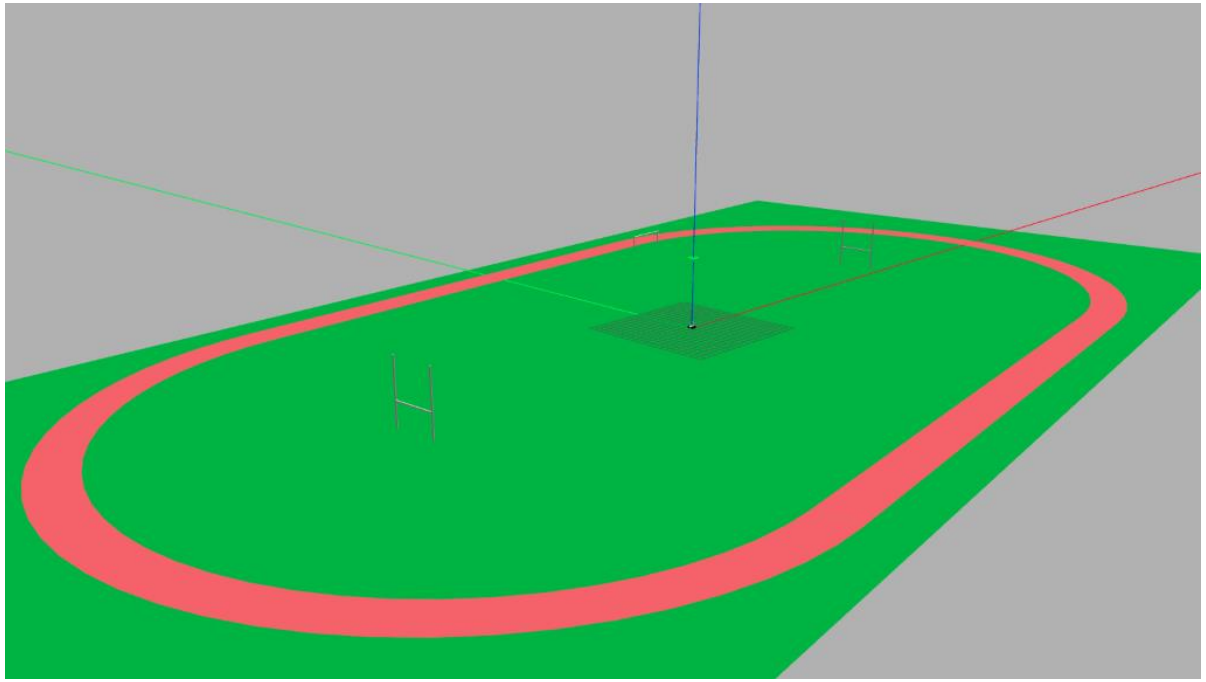
- Forme simple pour gazebo
- Capteurs simulés
- Codes réutilisables entre réel et simulation

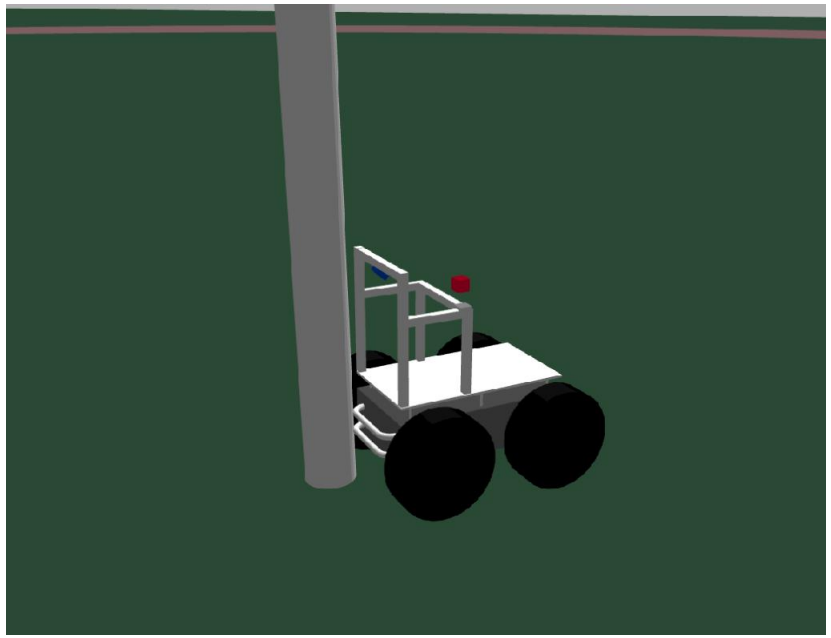


Simulation Saturne

Modélisation de l'environnement

- Environnement conforme à la réalité





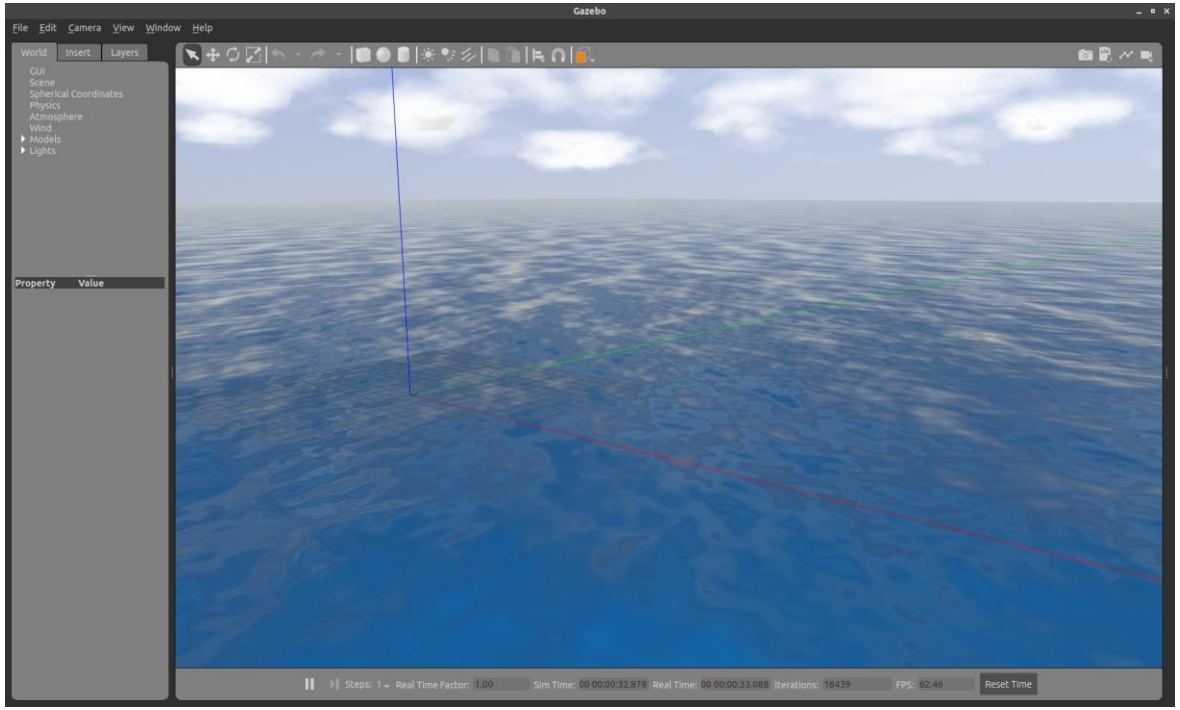
- Possibilité de tester les missions en avance

Simulation

UUV Simulator

- **VISUALISATION**
  - D'un monde marin
    - Fichier .world
  - D'AUV
    - Modèle 3D issu d'un logiciel de CAO

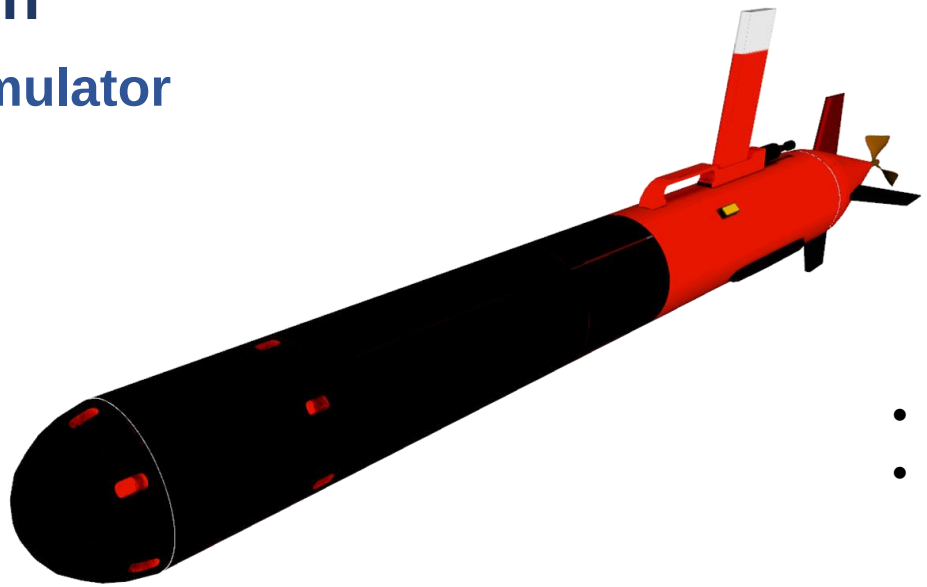
- **SIMULATION**
  - Des propriétés physiques du monde
    - Fichier .world
  - De la dynamique des AUV
    - Équation de Fossen
    - Fichiers de description .urdf ou .xacro
      - Dimensions de l'AUV
      - Inertie
      - Collisions
      - Capteurs
      - Actionneurs…



Exemple du monde « ocean_waves.world » sur UUV simulator

Simulation

UUV Simulator



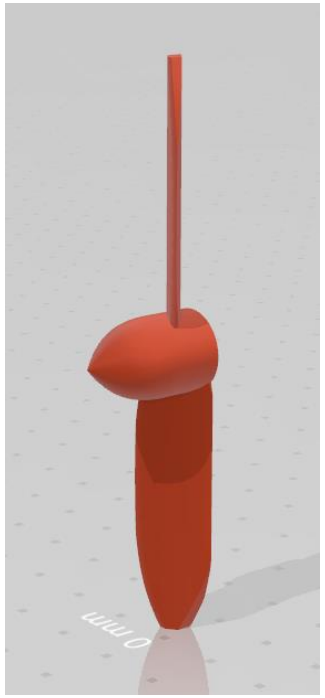
Modèle 3D du LAUV implémenté dans UUV simulator

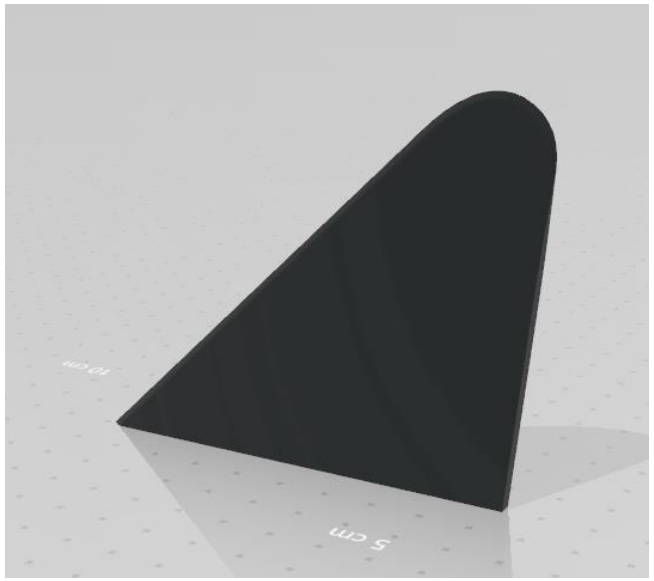
- Inspiration à partir de modèles d'AUV existants
- LAUV = *Light Autonomous Underwater Vehicles* développé par le "Laboratório de Sistemas e Tecnologia Subaquática (LSTS)" de l'Université de Porto et OceanScan-MST
  - AUV type torpille
  - 4 ailerons en "+" et un 5ème central
  - Propulseur à 3 pales à l'arrière
- Récupération des fichiers .urdf et .xacro et adaptation des paramètres pour le Riptide

Simulation

Robot Riptide

- Modélisation 3D





Simulation

Robot Riptide

- Entrée des paramètres du Riptide (masse, longueur, rayon…)
- Assimilé à un cylindre pour les paramètres hydrodynamiques
- Volume calculé pour que le robot soit équilibré
- Ajout et configuration des capteurs et actionneurs
  - GPS, IMU, Capteur de pression
  - 1 propulseur 2 pales, 3 ailerons

```
lauv_description/
    launch/
        upload.launch
    meshes/
        MBake_Complete.png
        README.md
        black_fin.dae
        body.dae
        body.stl
        propeller.dae
        red_fin.dae
        side_scan_sonar.dae
    robots/
        default.xacro
    test/
        test_urdf_files.py
        test_urdf_files.test
    urdf/
        actuators.xacro
        base.xacro
        gazebo.xacro
        sensors.xacro
        snippets.xacro

    CHANGELOG.rst
    CMakeLists.txt
    package.xml
```

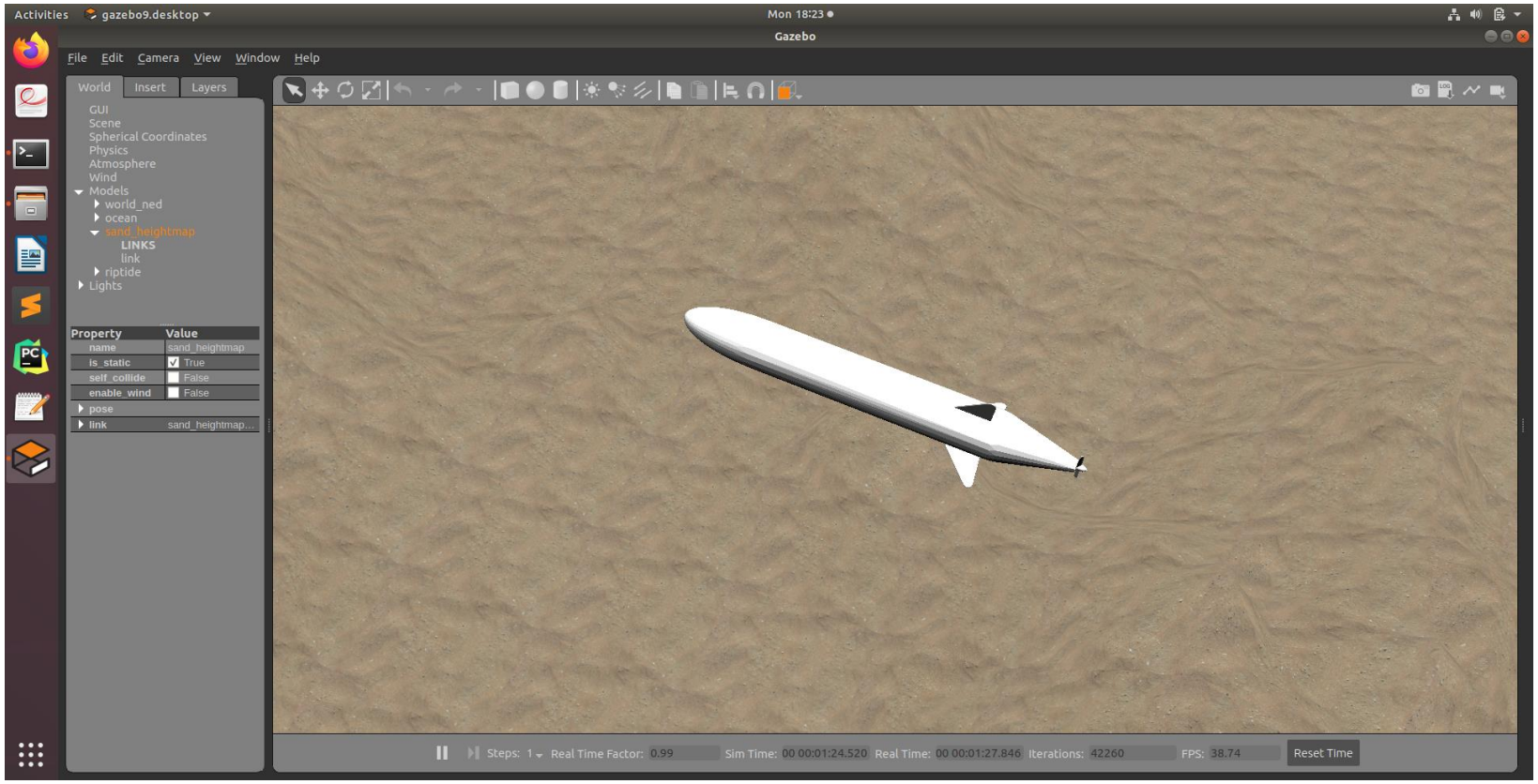
```
riptide_description/
    launch/
        launch_heading.launch
        launch_waypoint.launch
        upload.launch
    meshes/
        body.dae
        body.stl
        fin.dae
        propeller.dae
    robots/
        default.xacro
    src/
        remap.py
    test/
        test_urdf_files.py
        test_urdf_files.test
    urdf/
        actuators.xacro
        base.xacro
        gazebo.xacro
        sensors.xacro
        snippets.xacro

    CMakeLists.txt
    LICENSE
    NOTICE
    package.xml
    README.md
```

Simulation

Robot Riptide

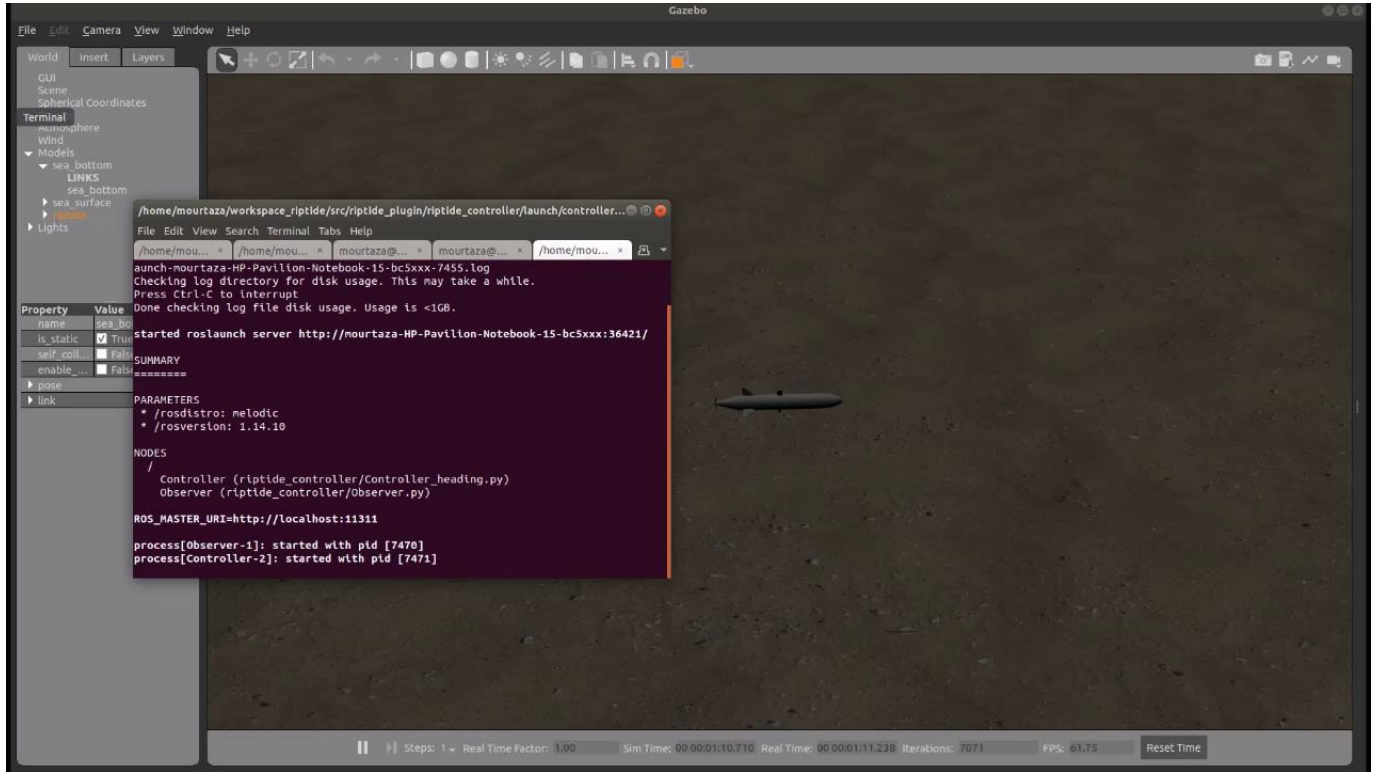
- Tests sur Gazebo avec rqt
- Monde ocean_waves



Simulation

Robot Riptide

- Interface avec le contrôleur
  - Contrôleur implémenté pour le robot réel
  - Ajout d'un nœud pour renommer les topics pour la simulation
- Tests de suivi de cap et de waypoints



Génération de trajectoires

Génération de trajectoires

Problématique

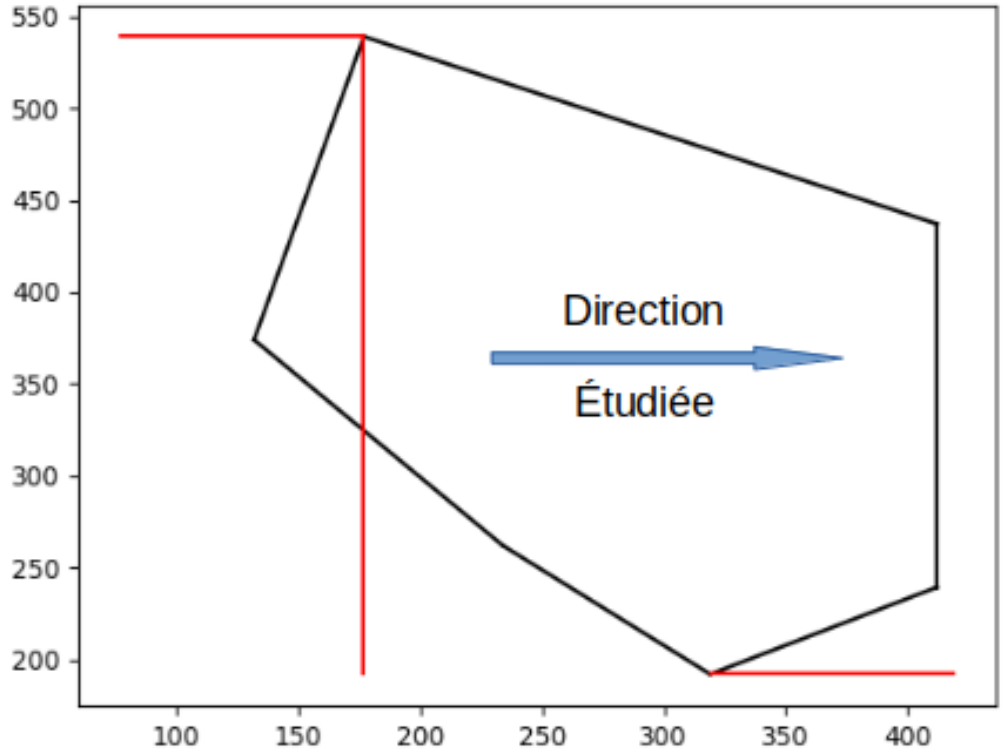
- Garantir l'évitement d'obstacle et le respect des limites de zone d'étude
- Explorer au maximum la zone d'étude
- Fournir une trajectoire gardant le câble tendu
- Fournir une solution indépendante de la taille du câble
- Minimiser la distance non utile parcourue

Génération de trajectoires

Boustrophédon sur un polygone convexe

Comment trouver la direction optimale de parcours ?

- Cas simple : polygone convexe
- Recherche sur les sommets uniquement
- Algorithme : *Rotating calipers*



Génération de trajectoires

Boustrophédon sur un polygone convexe

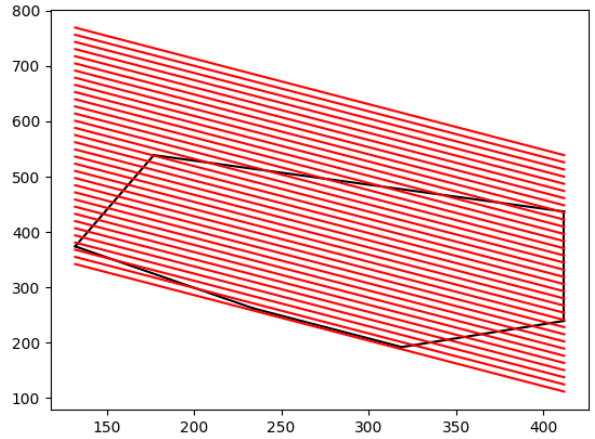
Comment générer les lignes du boustrophédon ?

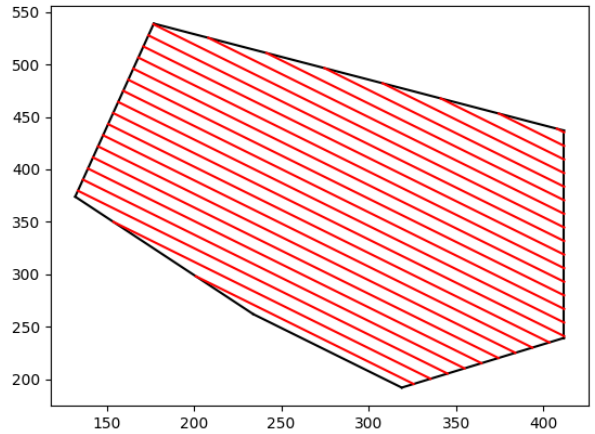
- Générer les droites parallèles

$$Si \quad |\alpha| < \frac{\pi}{2} - \delta a \quad : \quad y = ax + b + i_{itération}\,\delta b \quad où \quad a = \tan(\alpha), \quad b = y_{max} + |a\,x_{max}| \quad et \quad \delta b = \frac{d}{\sin\left(\frac{\pi}{2} - a\right)}$$

$$Sinon \quad x = x_{min} + i_{itération} * d$$

- Garder l'intersection des droites avec le polygone



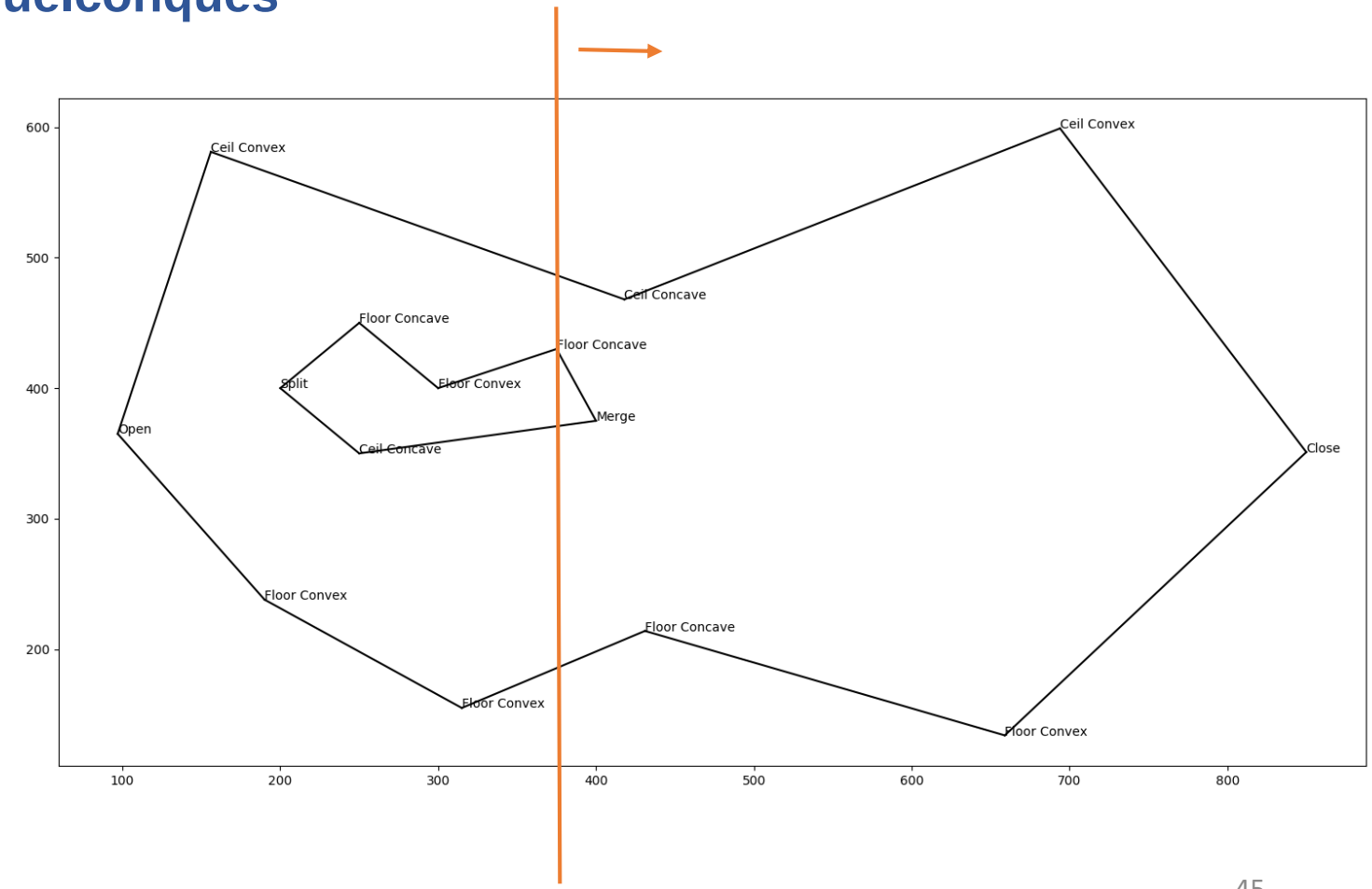


Génération de trajectoires

Généralisation aux polygones quelconques

Comment décomposer un polygone quelconque en sous-parties (cellules) convexes ?

- Algorithme : *Sweep line*
- Classer chaque sommet du champ de polygone : OPEN, SPLIT, MERGE, CLOSE, FLOOR/CEIL - CONVEX/CONCAVE
- Appliquer les actions correspondantes aux sommets identifiés

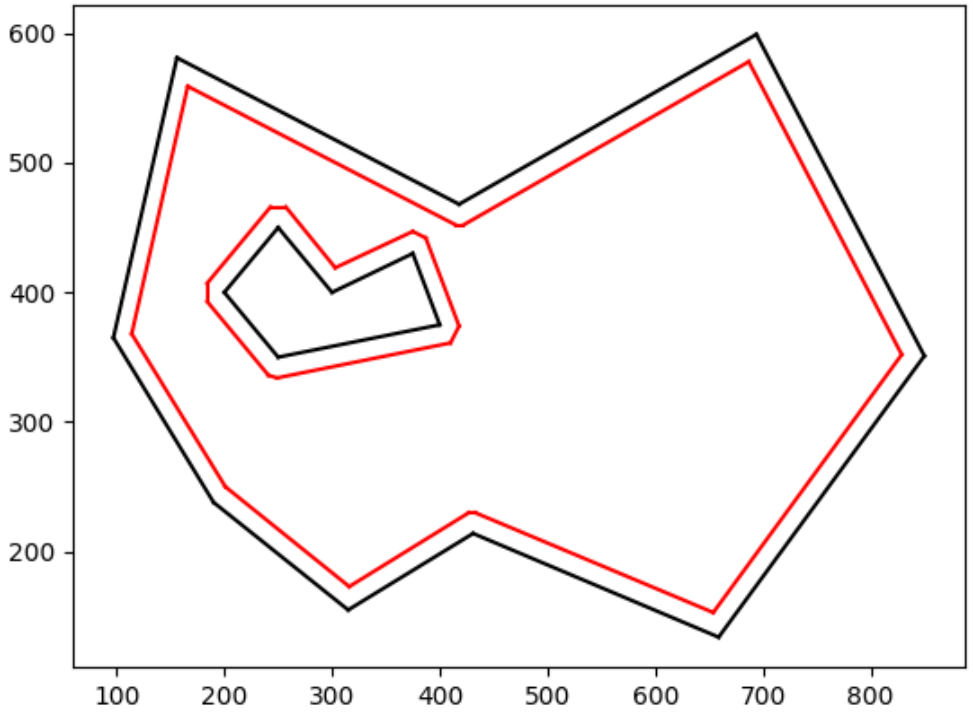


Génération de trajectoires

Prise en compte de la stratégie de virage

- Calculer le dépassement maximal R (avec un coefficient de sécurité S)
- Eroder le polygone extérieur d'un facteur R+S
- Dilater les polygones intérieurs d'un facteur R+S

= Problèmes de *polygon offset*



Génération de trajectoires

Ordre de parcours

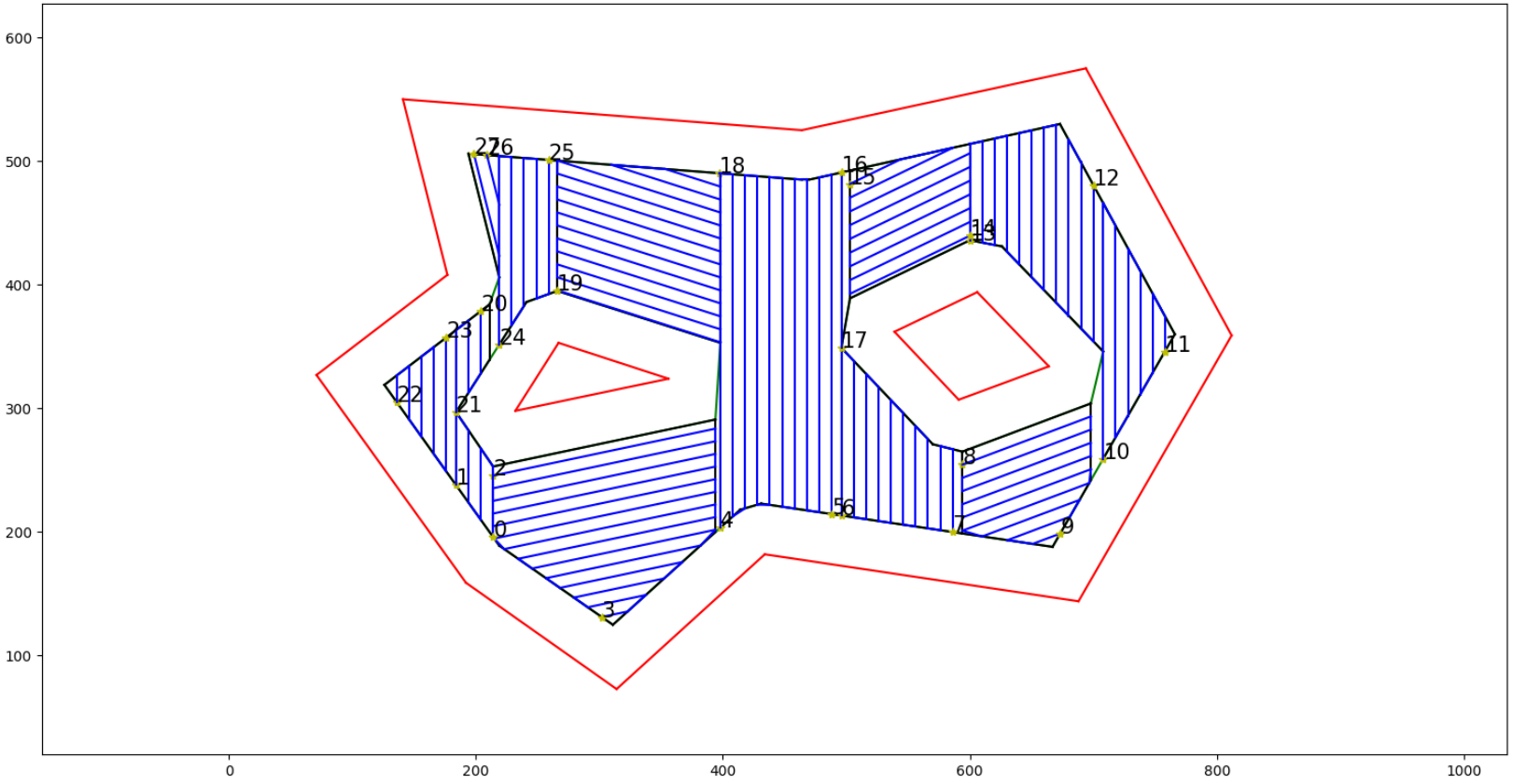
Comment diminuer la distance inutile parcourue ?

- Passer une seule fois par zone
- Passer dans toutes les zones

= Problème du voyageur de commerce

Génération de trajectoires

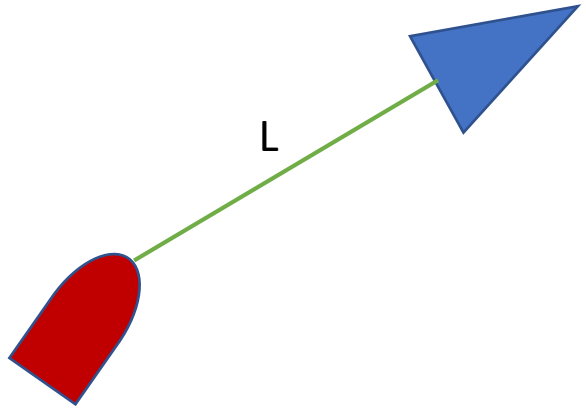
Résultat



Génération de trajectoires

Prévision de trajectoire

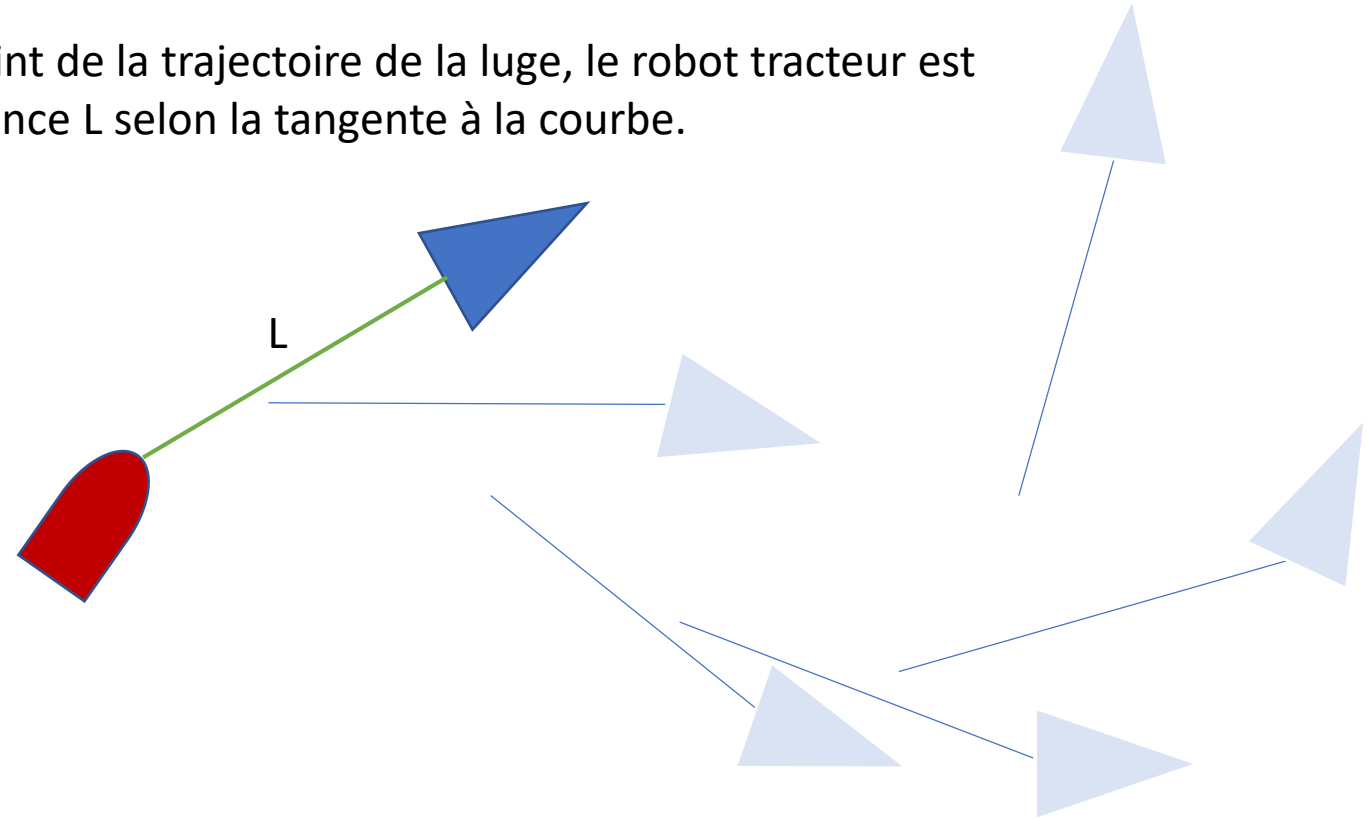
Comment faire suivre à la luge une trajectoire prédéfinie ?



Génération de trajectoires

Prévision de trajectoire

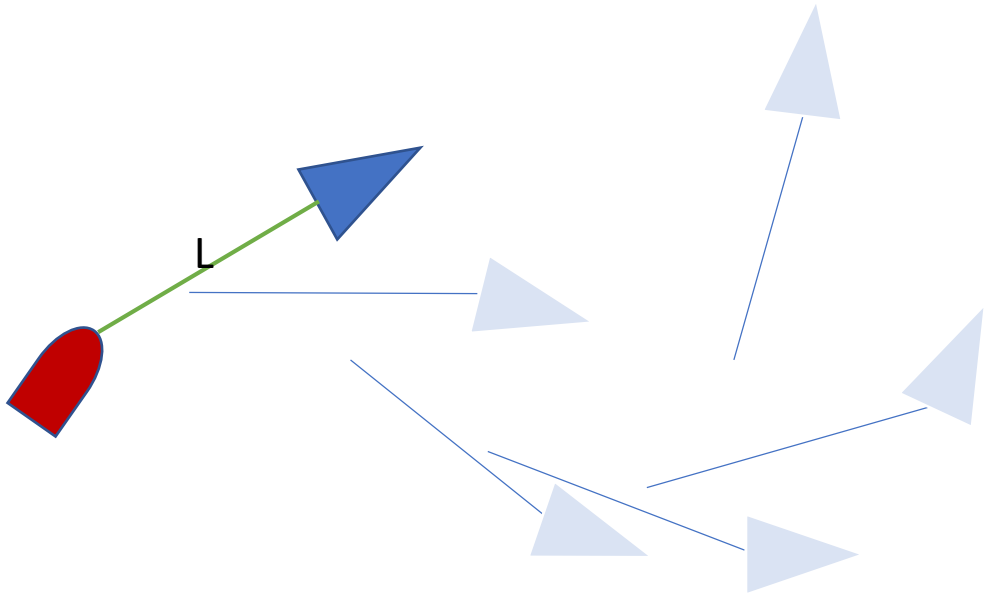
En tout point de la trajectoire de la luge, le robot tracteur est à une distance L selon la tangente à la courbe.

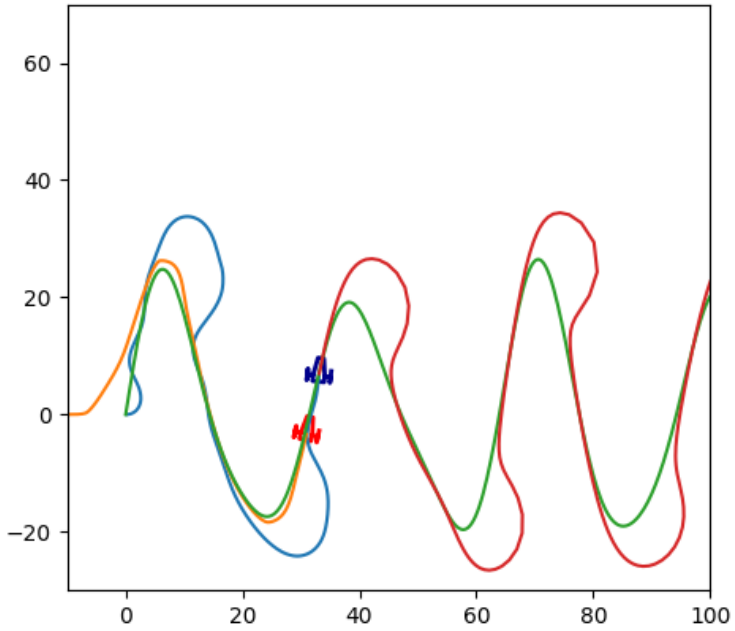


Génération de trajectoires

Prévision de trajectoire

On peut donc en déduire la trajectoire du robot.

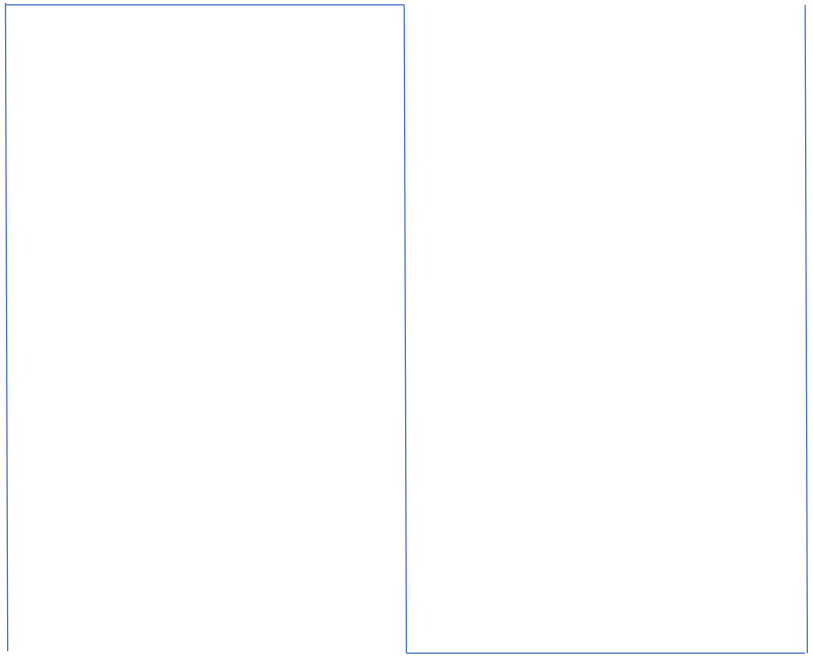




Génération de trajectoires

Prévision de trajectoire

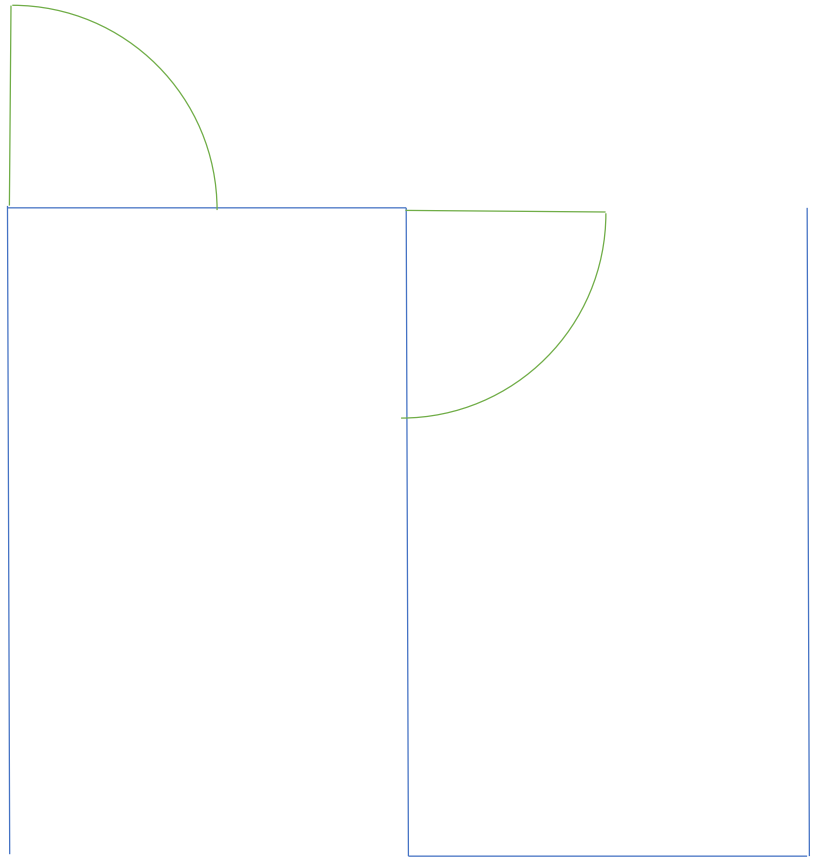
Simplification pour suivi de waypoints :



Génération de trajectoires

Prévision de trajectoire

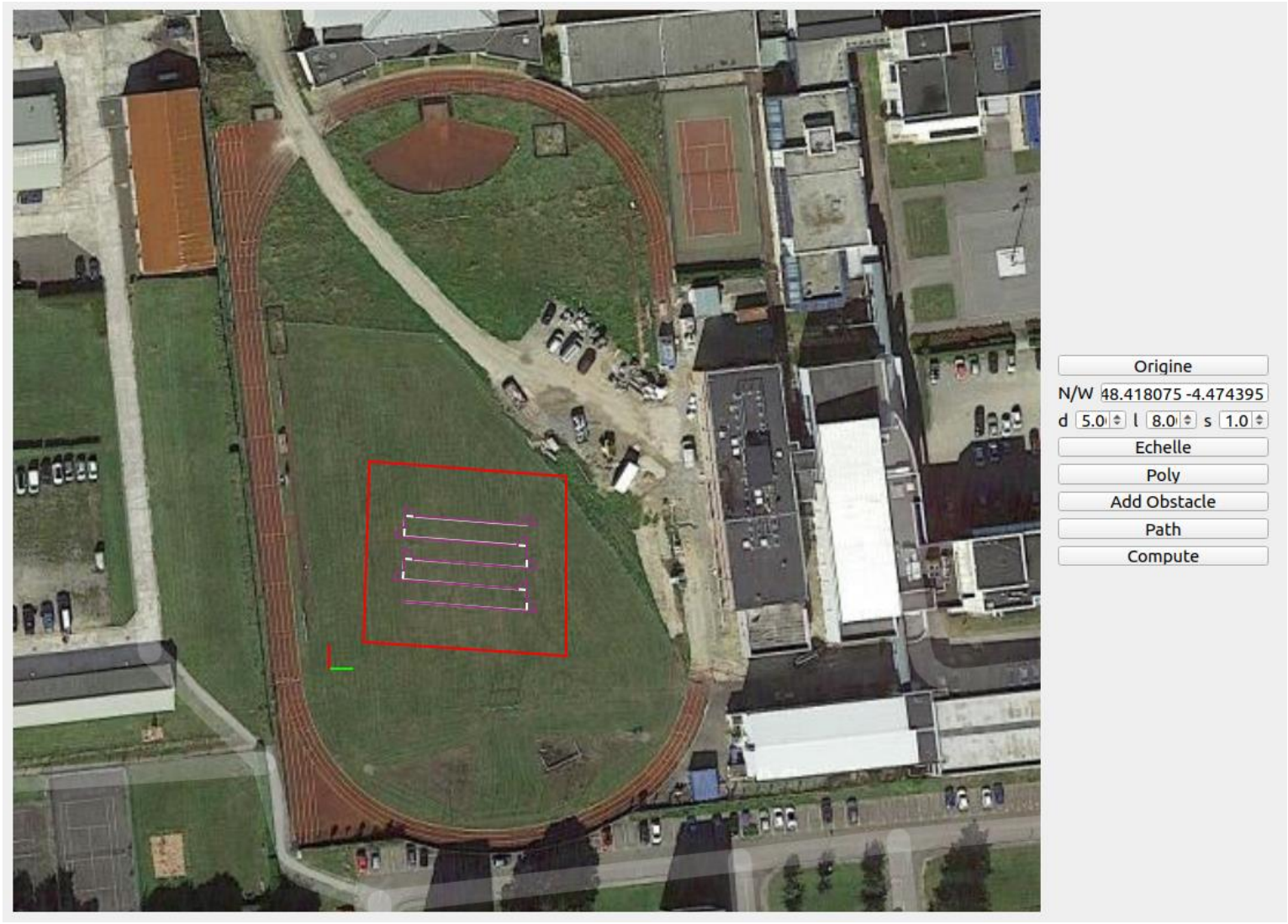
Simplification pour suivi de waypoints :



Génération de trajectoires

Interface utilisateur

Création d'une interface utilisateur pour simplifier la planification de mission



Post-processing des données

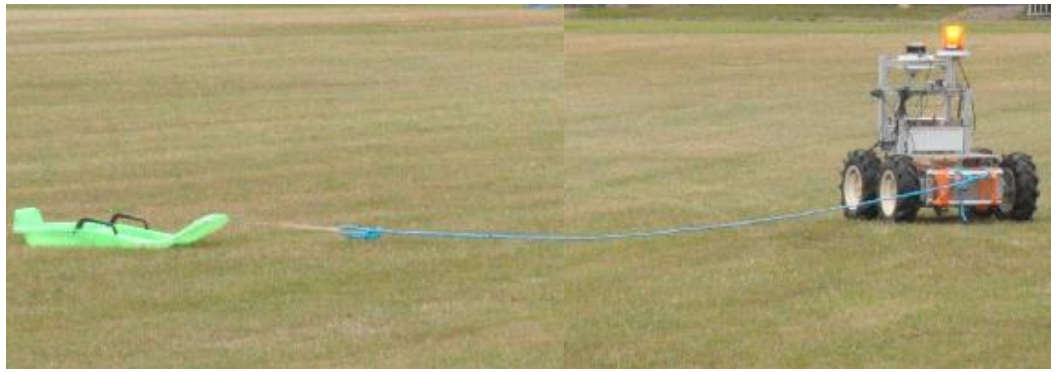
Post processing

Introduction

Objectifs :

- Estimer un ensemble fiable des trajectoires possibles prises par la luge en s'appuyant sur les capteurs de position embarqué sur Saturne

- Réaliser une carte fiable des zones ayant été, ayant peut-être été et n'ayant absolument pas été explorés par magnétomètre



$$[1] \quad \begin{cases} \dot{x} = f\big(x(t), u(t)\big) \ \ (evolution\ equation) \\ y^i = g\big(x(t_i)\big) \ \ (observation\ equation) \end{cases}$$

Equations d'état

Modélisation du système

Equations d'états

Modèle cinématique du robot Saturne tractant la luge :

$$\dot{x} = f(x,u) = \begin{cases} \dot{x_1} = \dfrac{u_1+u_2}{2}.cos(x_3) & \textbf{(1)} \\ \dot{x_2} = \dfrac{u_1+u_2}{2}.sin(x_3) & \textbf{(2)} \\ \dot{x_3} = u_2 - u_1 & \textbf{(3)} \\ \dot{x_4} = -\dfrac{u_1+u_2}{2}.sin(x_4) - \dot{x_3} & \textbf{(4)} \end{cases}$$

- *x_1, x_2 et x_3 sont respectivement l'abscisse, l'ordonné et l'orientation du robot de tractage*
- *x_4 est l'angle de la luge avec le robot.*

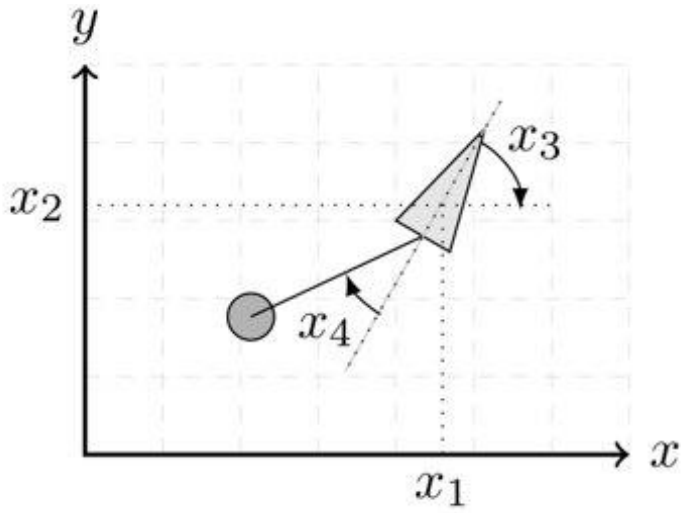
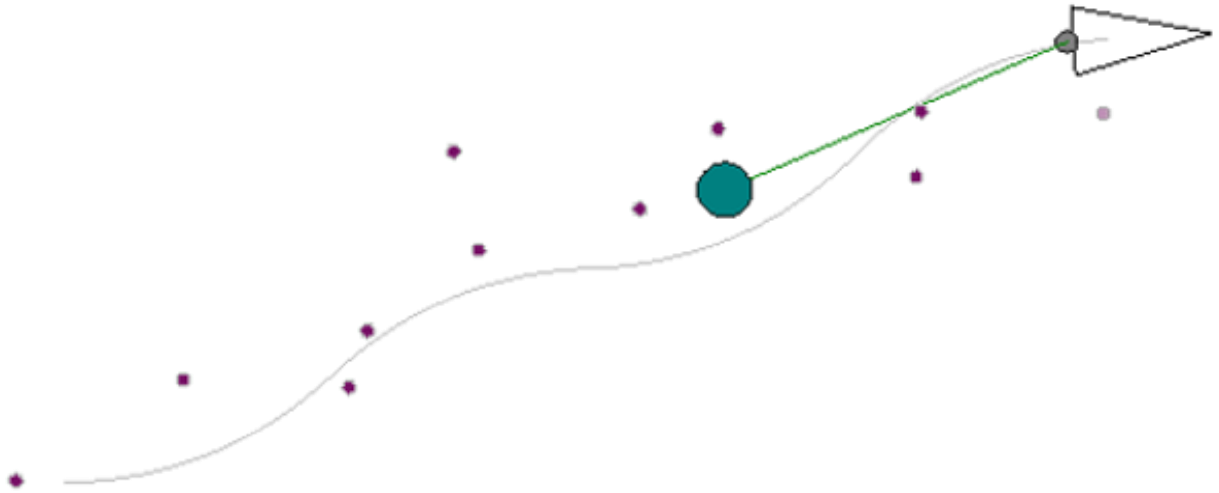


Fig. 1. Diagram representing system state variables

Modélisation du système

Simulation



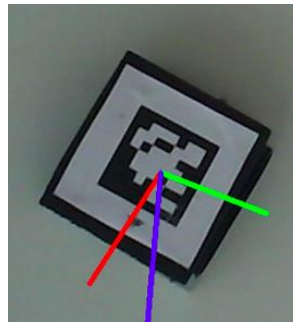
Simulation VIBES

- VIBES
- Intègre l'équation d'évolution
- Méthode d'intégration d'Euler
- GNSS simulé
- Boussole simulée

Angle de la luge

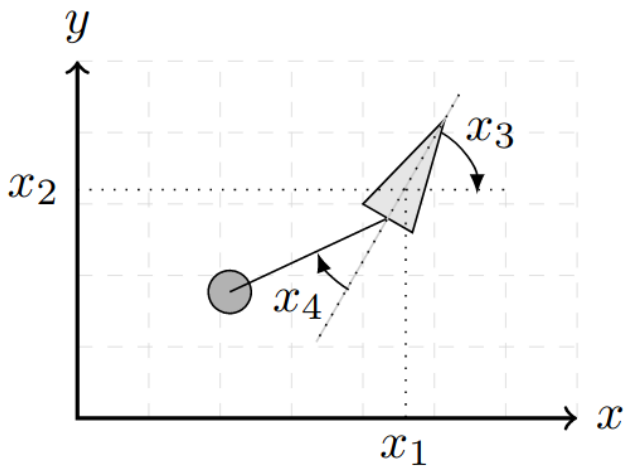
Obtention par une caméra

- Estimation de la position avec des ArUco.
- Algorithme de traitement d'images -> superposition de ligne.



Obtention par une équation d'état

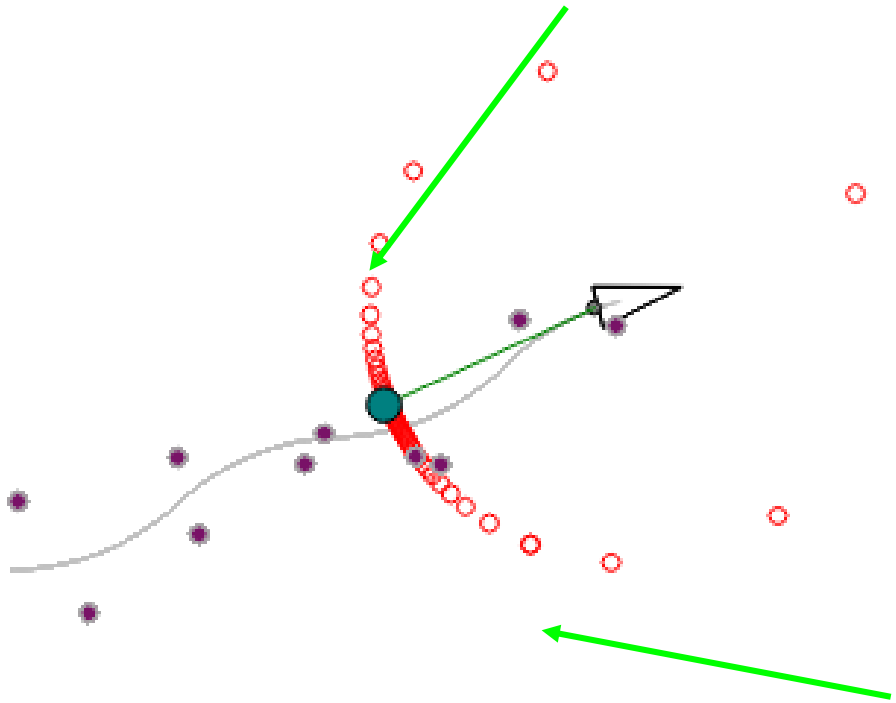
$$\dot{\mathbf{x}}_4 = -\frac{\mathbf{u}_1+\mathbf{u}_2}{2}.sin(\mathbf{x}_4) - \dot{\mathbf{x}}_3$$



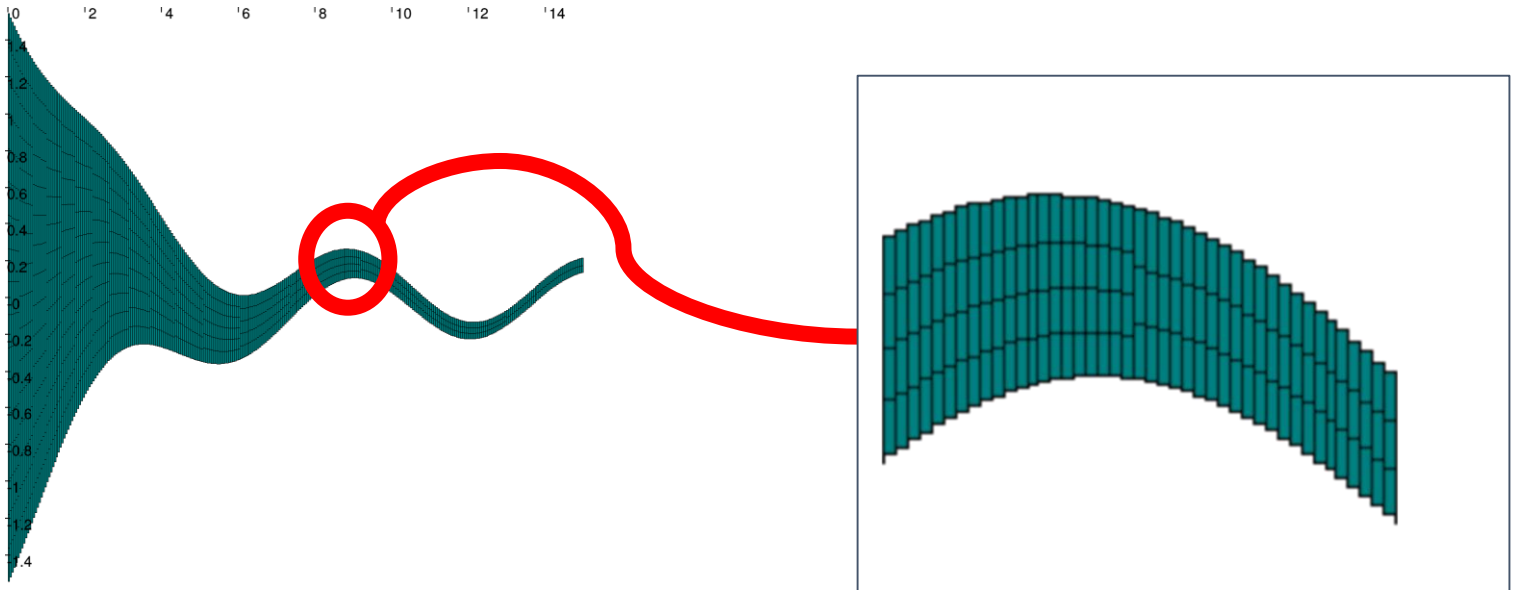
Vérification du modèle cinématique par méthodes statistiques

Méthode de Monte-Carlo

- Génération d'un ensemble de conditions initiales d'angles de luge
- Intégration de l'équation différentielle



Vérification du modèle cinématique par méthodes ensemblistes

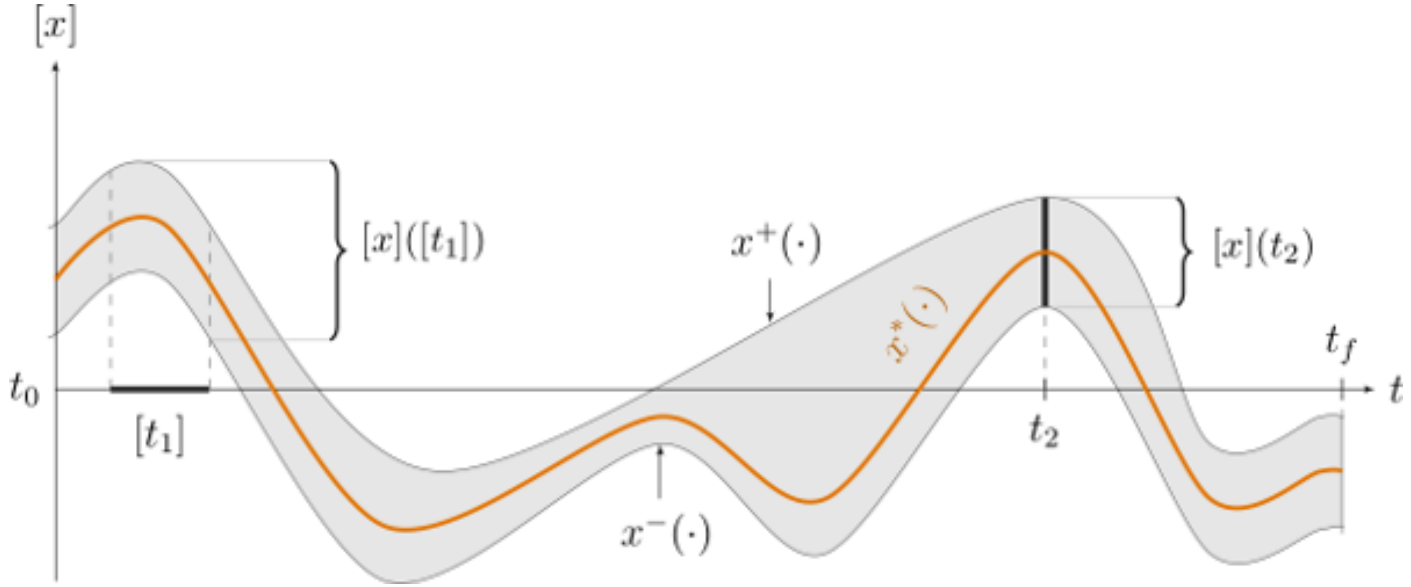


Méthode ensembliste : analyse par intervalle

- Certitude : aucune condition initiale oubliée.
- Similairement à la méthode Monte-Carlo : on a une convergence de la solution autour de l'angle de la luge.

Localisation par approche ensembliste

Tubes



- Objet permettant d'encadrer toutes les trajectoires possibles d'un système.
- Ensemble d'intervalles évoluant au cours du temps.

Localisation par approche ensembliste

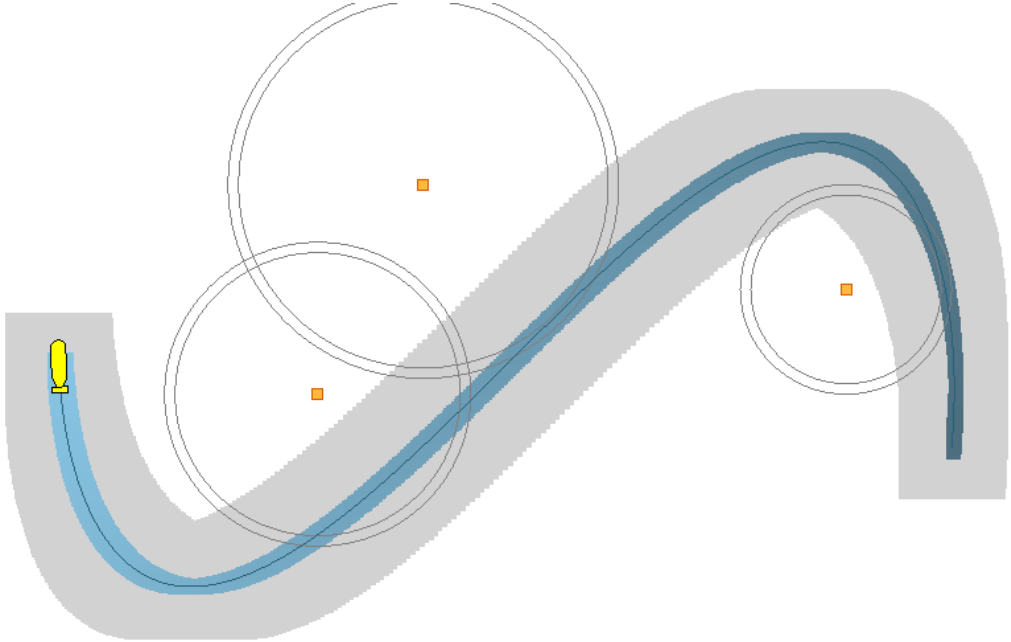
Réseau de contracteurs

CtcDeriv: $\dot{x}(t) = v(t)$

$$\dot{x}(\cdot) = v(\cdot) \longrightarrow \mathcal{C}_{\frac{d}{dt}}\left([x](\cdot), [v](\cdot)\right)$$

CtcEval: $y_i = x(t_i)$

$$\left.\begin{array}{l} y_i = x(t_i) \\ \dot{x}(\cdot) = v(\cdot) \end{array}\right\} \longrightarrow \mathcal{C}_{\text{eval}}\left([t_i], [y_i], [x](\cdot), [v](\cdot)\right)$$



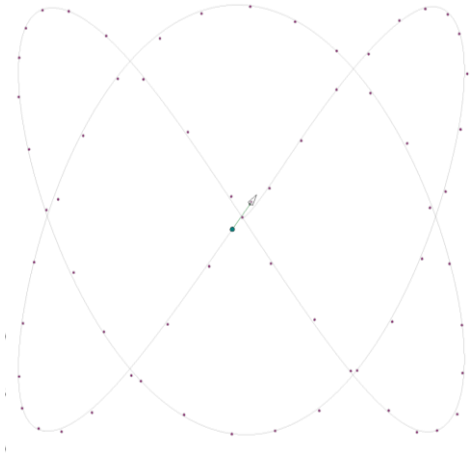
Localisation par approche ensembliste

Localisation du MagMap

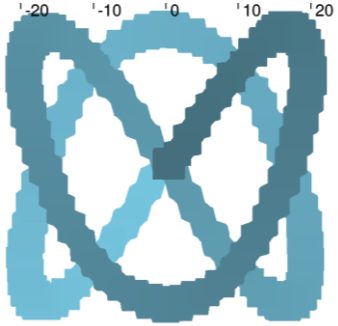
Décomposition des contraintes:

$$\mathbf{v}(\cdot) = f(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$$
$$\dot{\mathbf{x}}(\cdot) = \mathbf{v}(\cdot)$$
$$\dot{\mathbf{v}}(\cdot) = \mathbf{a}(\cdot)$$
$$\mathbf{d}(\cdot) = \mathbf{x}_{1,2}(\cdot) - \mathbf{x}_{\mathbf{m}}(\cdot)$$
$$\mathbf{t} = \mathbf{x}_3 + \mathbf{x}_4$$
$$\mathcal{L}_{polar}\left(\mathbf{d}(\cdot),\ [L],\ \mathbf{t}\right)$$
$$\mathcal{L}_{eval}\left(t_i,\ [\mathbf{y}_{1,2}],\ \mathbf{x}_{1,2}(t_i),\ \mathbf{v}_{1,2}(t_i)\right)$$
$$\mathcal{L}_{eval}\left(t_j,\ [\mathbf{y}_{4,5}],\ \mathbf{v}_{1,2}(t_j),\ \mathbf{a}_{1,2}(t_j)\right)$$

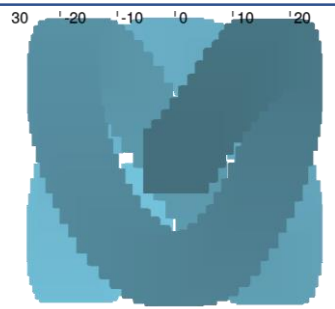
Trajectoire



Tube saturne



Tube magnétomètre

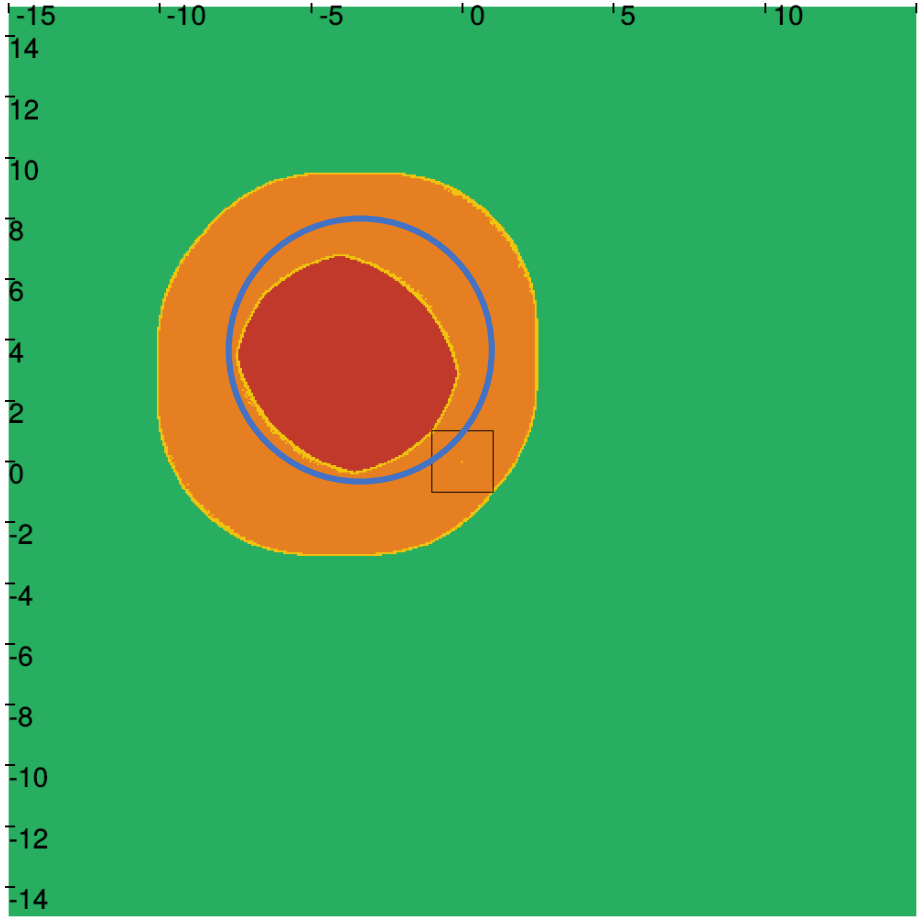


Couverture de la cartographie

Thicksets

- Connaissance boite contenant le robot tracteur
- Encadrer ensemble mesuré (cercle bleu) par Subset (zone rouge) et Supset (zone orange)

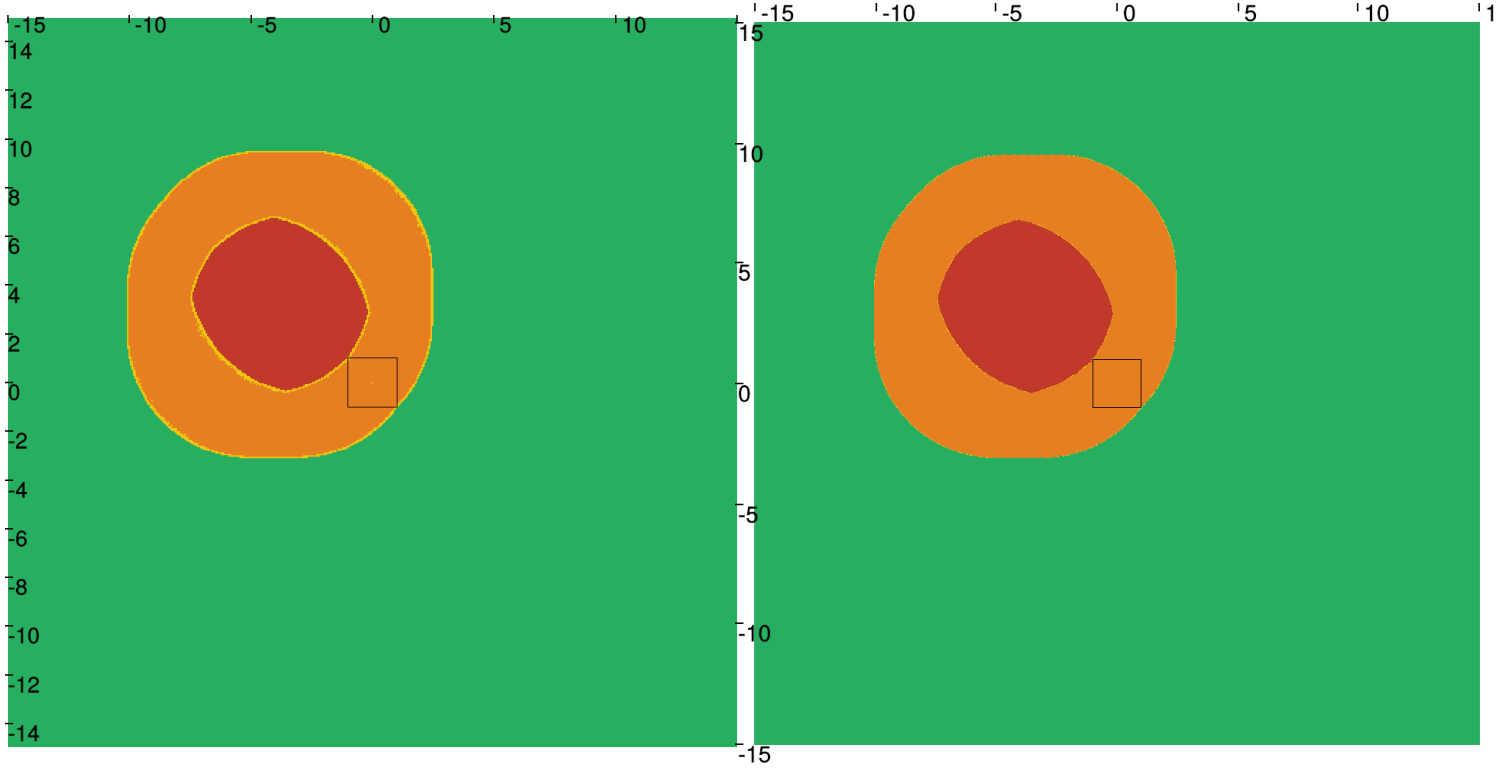
- Utilisation Thicksets, librairie d'Ibex



Couverture de la cartographie

SIVIA

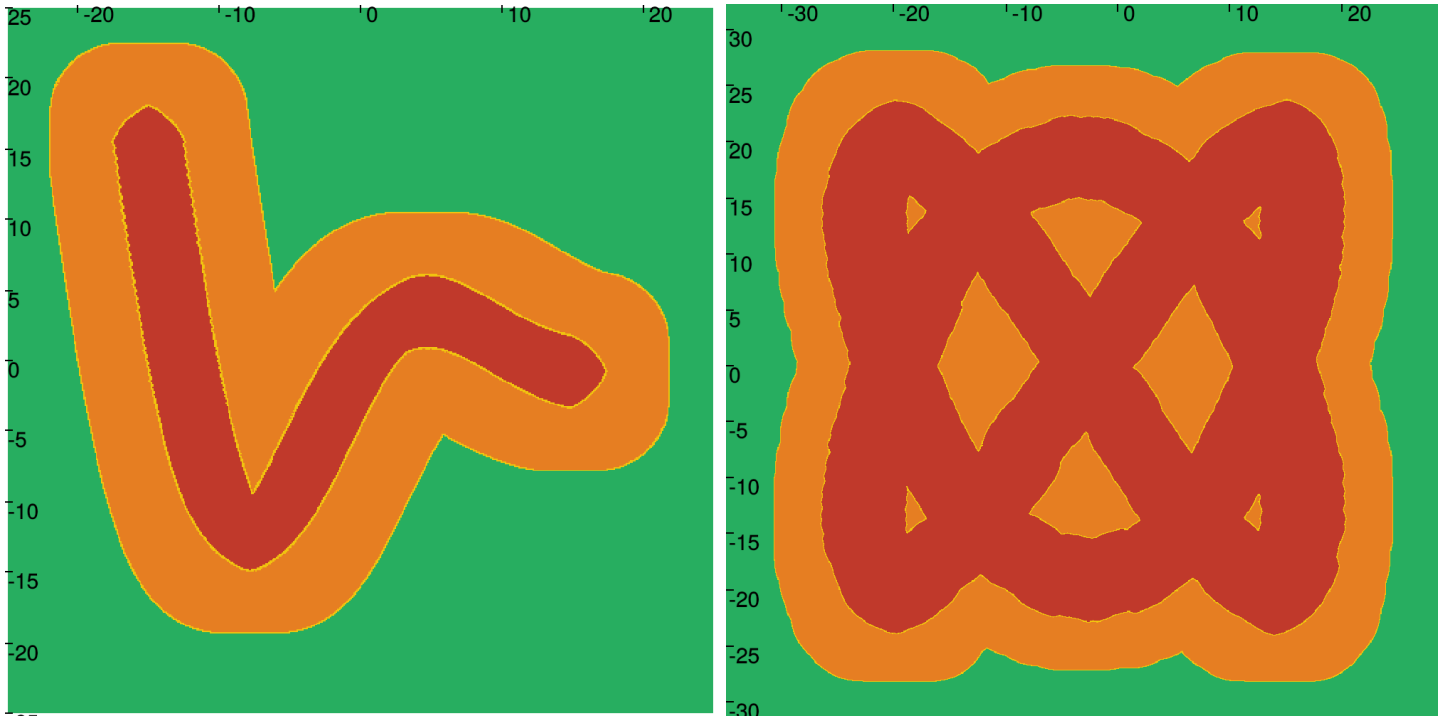
- Algorithme inversion ensembliste SIVIA
- Recursif condition d'arrêt sur taille boite (aire jaune)
- Sépare les domaines :
  - IN : Rouge
  - OUT : Orange
  - EMPTY : Vert
  - UNKNOWN : Jaune



Couverture de la cartographie

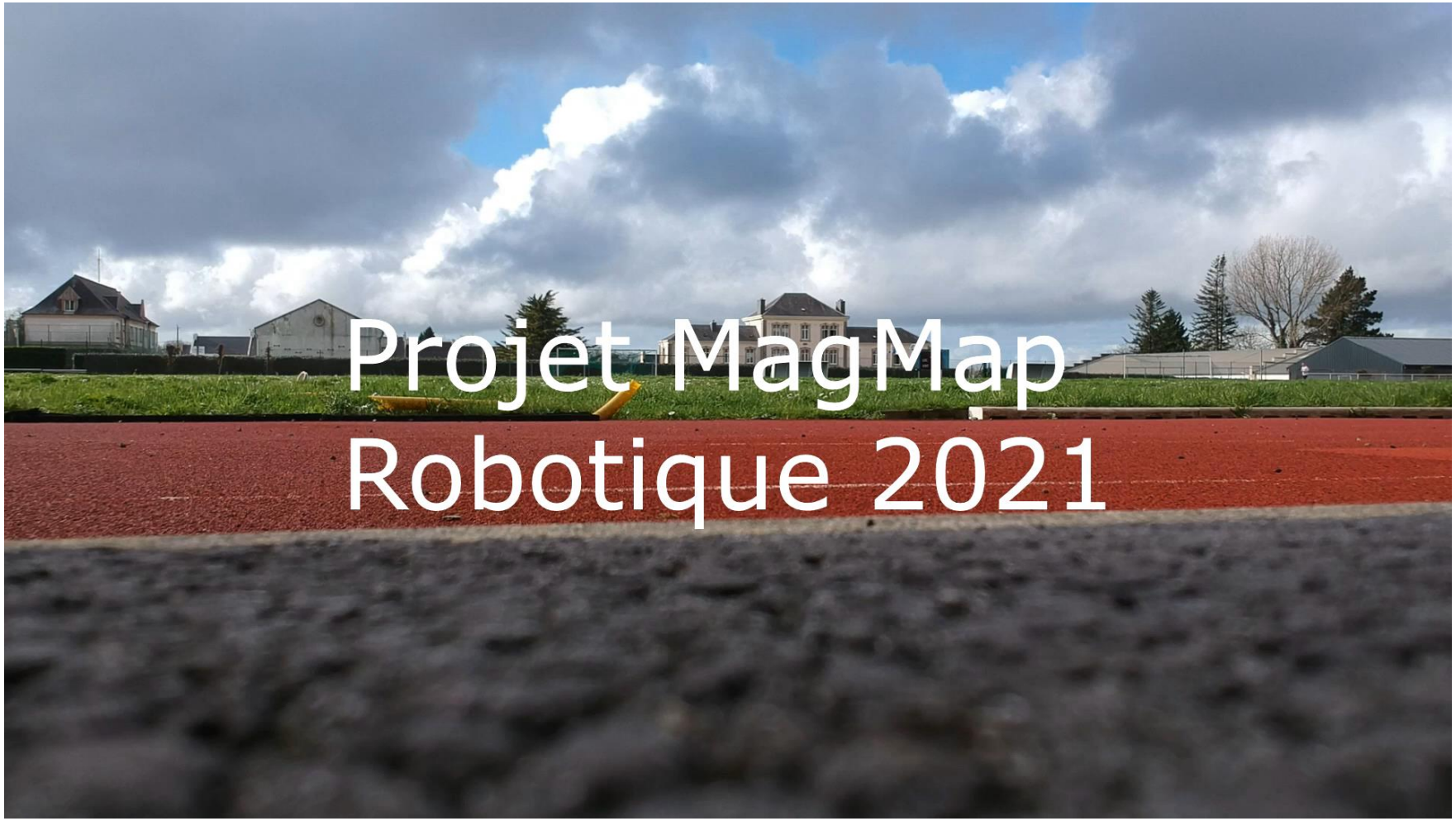
Couverture

- Union ensembles le long de la trajectoire
- Obtient couverture cartographie
- Prends du temps de calcul, d'où post-processing

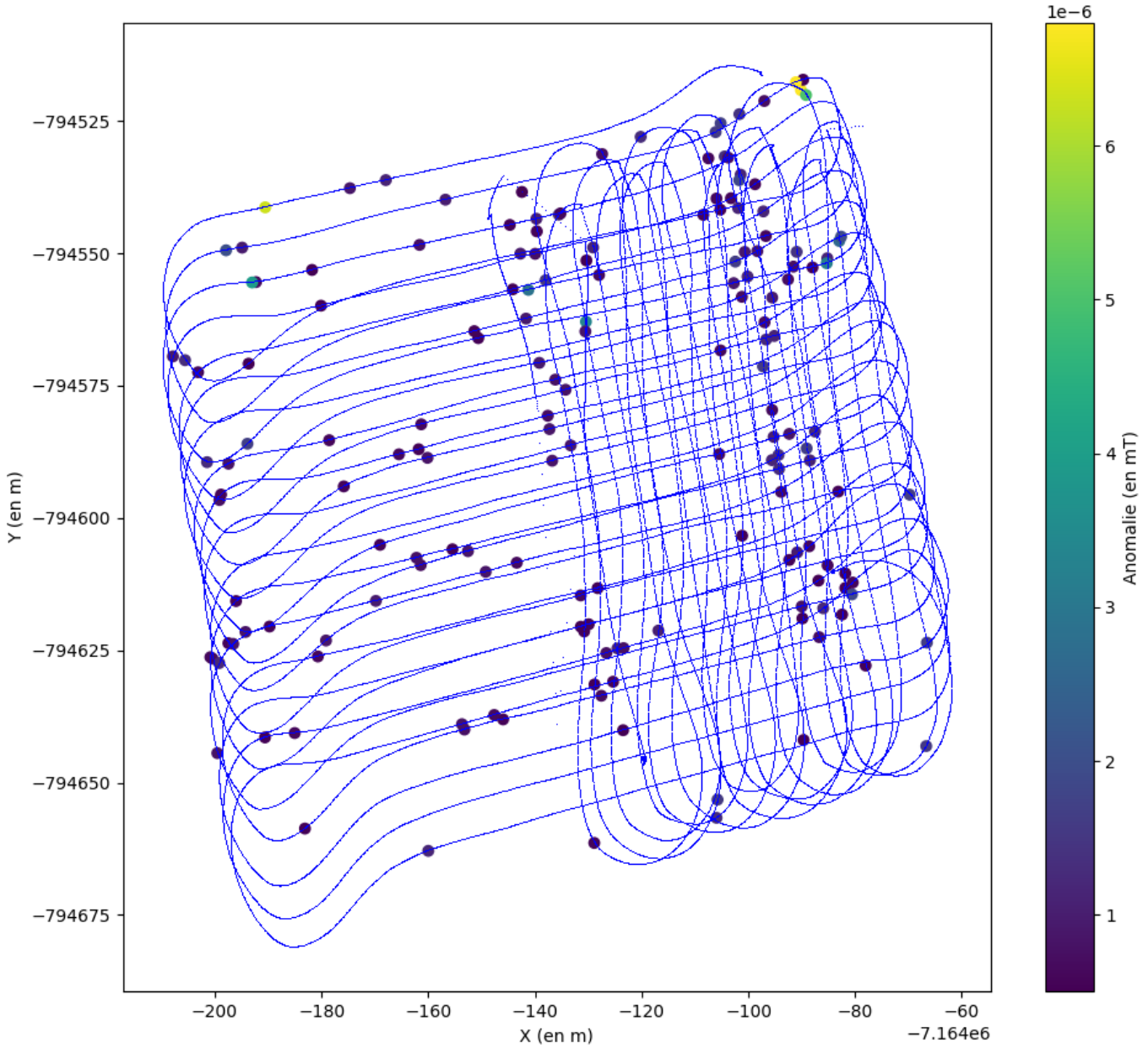


Expérimentations et résultats

Expérimentations



Résultats


1e−6
−794525
−794550
−794575
−794600
−794625
−794650
−794675
Y (en m)
−200
−180
−160
−140
−120
−100
−80
−60
X (en m)
−7.164e6
6
5
4
3
2
1
Anomalie (en mT)

Conclusion