Auteur: Abdelghani GRICHE
Master II PRO CSNIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Stage de fin d'études : APPLICATION DE DEPLACEMENT AUTOMATIQUE POUR CYCAB

UNIVERSITE JOSEPH FOURIER
SCIENCES.TECHNOLOGIE.SANTÉ

Remerciements

Je tiens à remercier toute l'équipe SED en particulier Nicolas Turro , mon tueur , Roger Pissard-Gibolet , Soraya Arias , Jean Francois Cuniberto, Christophe Braillon , Gerard Baille ...
pour le projet de stage , le cadre et l'ambiance de travail qui m'ont offert.

1-introduction

Contexte

Le stage se déroule au sein du service Support Expérimentations et Développements logiciels (SED) de l'INRIA Rhône-Alpes dont un des rôles est la mise en œuvre des outils matériels et logiciels pour les expérimentations robotiques des projets de recherche du site. Le robot Cycab est un véhicule automobile autonome de la taille d'une voiture sans permis. Il est utilisé par les chercheurs de l'équipe-projet e-motion pour tester les applications de leurs travaux de recherche. Plusieurs résultats ont été obtenu, notamment dans le domaine de la planification de trajectoire, de la localisation, de la modélisation de l'environnement et de la navigation réactive.

De nombreuses évolutions ont été amenées sur le Cycab : mise à jour matérielle, refonte du logiciel embarqué, intégration de nouveaux capteurs comme le GPS centimétrique.

Objectifs du projet/stage

L'objectif du stage est de réaliser une application de déplacement automatique en intégrant les évolutions du Cycab.

Je devrai étudier et implémenter un algorithme de suivi de chemin prédéfini en contrôlant la vitesse d'avancement du véhicule et son cap. Un retour d'information concernant la position du véhicule sera obtenu grâce au GPS ou à un calcul odométrique.

En fonction des résultats, plusieurs approches seront envisagées, plus ou moins simples : follow the carrot, pure pursuit ; ou plus sophistiquées (contrôle de systèmes chaînés).

L'implémentation s'inscrira dans une architecture logicielle *maison* en C++ dédiée a la programmation des robots mobiles.

Les premiers tests, ainsi qu'une première estimation des gains ou paramètres de l'algorithme s'effectueront sur un simulateur de véhicule.

Enfin, nous procèderont à la mise au point finale sur le véhicule Cycab.

Plan du stage

Dans la première partie , je présenterais le cycab , et le GPS qui sont les parties physiques (actionneur,capteur) essentielles au test réel de l'algorithme.
Puis, je ferais un rappel sur l' état de l'art concernant les différents techniques de suivi de chemin.
Dans la seconde partie, des lois de commande sera établie a partir de la théorie des systèmes chainées.
La 3 ème partie traitera du filtrage Kalman appliqué au capteur GPS et des méthodes d'interpolation pour la détermination des différentes variables tel l'erreur latérale ,abscisse curviligne ,et courbure.

Environnement de travail :

je dispose d'un Pentium 4 configuré sous LINUX RED HAT comme système d'exploitation afin de afin de mieux se rapprocher de la configuration de l'OS du PC embarque du cycab.

Matériels et ressources mis à disposition :

- du logiciel SCILAB pour l'exploitation des calculs complexes et l'affichage des graphes .
- bibliothèque INRIA & bibliographie concernant **l'automatique avancée** (articles scientifiques IEEE internet, ressources INRIA)
- le simulateur MgEngine(p12)

SOMMAIRE

2-Présentation de l'INRIA

2.1)Généralités :

Adresse :
INRIA RhôneAlpes
Inovallée
655 Avenue de l'Europe
Montbonnot
38334 Saint Ismier cedex
France

Téléphone :
+33 4 76 61 52 00

2.2)historique :

Le centre de recherche INRIA Grenoble - Rhônes-Alpes a a été crée en 1992 et rassemble aujourd'hui environ 500 personnes. Ses 26 équipes-projets de recherche se répartissent entre Montbonnot, près de Grenoble, L'ENS à Gerland et le campus universitaire de la Doua à Lyon. Elles sont pour la plupart communes avec des établissements locaux : universités de Grenoble et Lyon (universités Joseph-Fourier, Institut national polytechnique de Grenoble, université Claude-Bernard), L'école normale supérieure de Lyon et l'INSA de Lyon,ainsi que le CNRS, et plus précisément avec leurs laboratoires LIG, LJK, LIP ou CITI. Le centre entretient par ailleurs de nombreuses relations avec les grands acteurs locaux comme STMicroelectronics, France Telecom, Xerox, et irrigue le tissu industriel avec les nombreuses start-up qui ont vu le jour dans ses murs. Ses équipes participent aux pôles de compétitivités régionaux Imaginove et Minalogic et aux pôles mondiaux pôleAESe et System@tic. Sur le site lyonnais, les équipes du centre participent au RTRA Innovations en infectiologie. Les trois grandes priorités thématiques du centre sont : maîtrise des ressources dynamiques et hétérogènes des systèmes embarqués aux infrastructures de calcul et de communication " ; modélisation et simulation des phénomènes multiéchelles et multicomposants ; perception et interaction avec des environnements réels et virtuels.

Direction:

François Sillion, directeur du centre de recherche INRIA Grenoble - Rhône-Alpes depuis le1er février 2007.

2.3-Service Expérimentations et Développement logiciel SED

rôle :

Le service Support Expérimentations et Développements logiciels (SED) a pour rôle la mise en œuvre des outils matériels et logiciels pour les expérimentations robotiques des projets de recherche du site.

Le service gère 3 plateformes :

2.3.1)Les plates-formes Robotique et Vision

2.3.1.1)La Halle robotique

La Halle robotique contient les plates-formes Robotique et Vision:

2.3.1.2)Orccad: Un contrôleur robotique pour les plates-formes

Orccad est un outil d'aide à la conception de contrôleur robotique développé par l'INRIA et utilisé pour le robot Bip, le premier prototype du Cycab et le RX90.

2.3.1.3)Les Véhicules électrique Automatiques Cycab.

2.3.1.4)La bio-robotique: marche artificielle

2.3.2)Plate-forme de Réalité-Virtuelle:

La plate-forme (160 m²) est basée sur un couple de superculateurs graphiques (SGI), et se compose d'une salle d'immersion plein-pied (grand écran cylindrique) , d'un plateau de prise de vue avec fond bleu et d'une salle de manip.

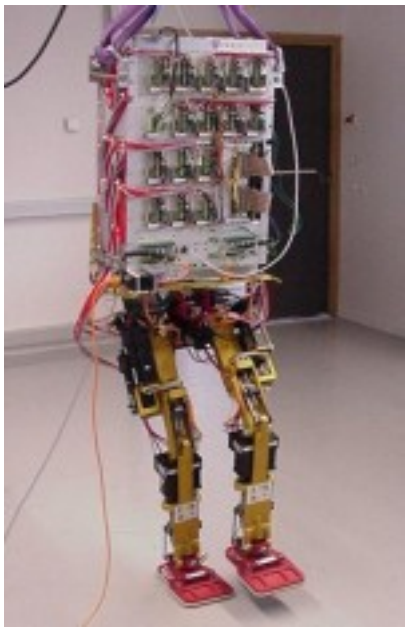
2.3.3)Les plates-formes GRAPPES

Le service supporte deux plates-formes de Grappes expérimentales :

=>Une grappe de PC connecté sur un mur d'images interactif

=>Une grappe d'Itanium2 :

cycab



Robot BIP

1.Présentation du cycab

Le Cycab

Le Cycab a la forme d'une voiture pouvant transporter deux passagers. Dans le but de se déplacer en zone urbaine, son gabarit est assez petit (longueur : 1,90 m, largeur : 1,20 m). Il ne pèse que 300 kg, ce qui lui permet d'avoir une autonomie de deux heures. Il est équipé de six moteurs électriques : un pour chaque roue du véhicule et un pour activer le vérin de chaque direction (avant et arrière). Sa vitesse maximale est de 18 km/h.

Hardware du cycab

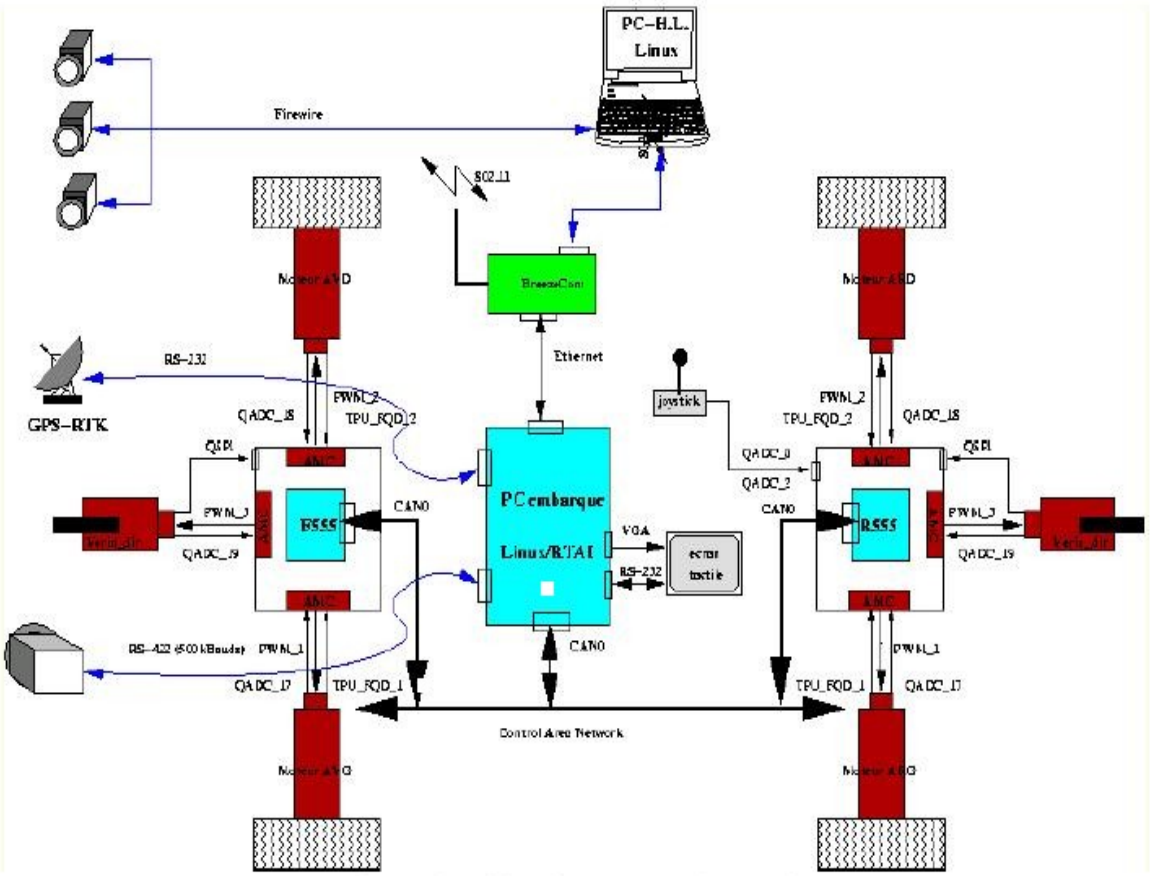


Schéma détaillant les organes du Cycab (source INRIA)

Il dispose de différents capteurs pour permettre sa localisation : télémètre, GPS centimetrique(p8), camera ,odométrie,Un PC embarqué enregistre et gère toutes les informations des capteurs et des actionneurs du robot cycab en temps réel .(voir ANNEXES)
Sa double architecture matérielle et logicielle (p10)lui permet de rendre portable toute implémentation écrite en C++ et a condition que la machine dédié au test doivent avoir les mêmes caractéristiques (CPU,noyau linux)

3-Présentation du GPS centimétrique Thalès ZMAX

Le capteur principal pour mon projet est le GPS centimétrique Thalès ZMAX.Son utilisation est assez complexe car elle nécessite une programmation et une initialisation couteuse en temps

3.3.1)module GPS ZMAX

Il est constitué de deux modules GPS composes des antennes GPS, de calculateurs, des batteries avec leurs chargeurs , d'un émetteur et un récepteur radio,et des trépieds pour leur positionnement sur le terrain.

Pour faire fonctionner en mode RTK ,il faut un récepteur GPS fixe appelé « base » dont la position en coordonnées GPS est connue, ainsi qu'un autre appelé « mobile » pour prendre les mesures GPS.

photo: GPS Mobile

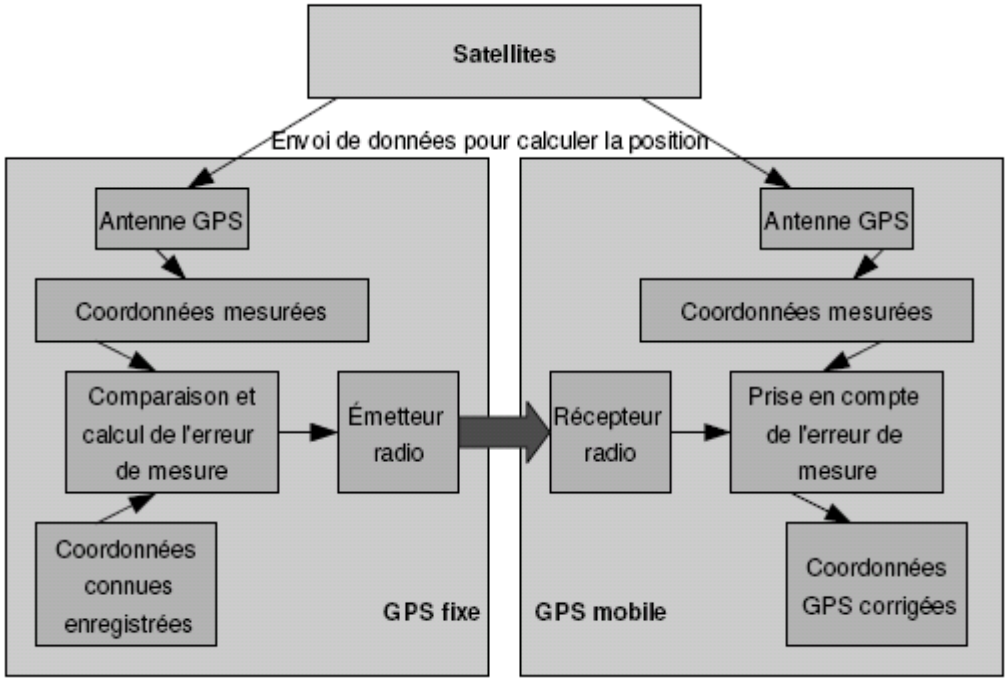
photo:GPS mobile(à gauche) et GPS fixe (à droite)

3.3.2)Fonctionnement du système RTK (real time Kinematic)

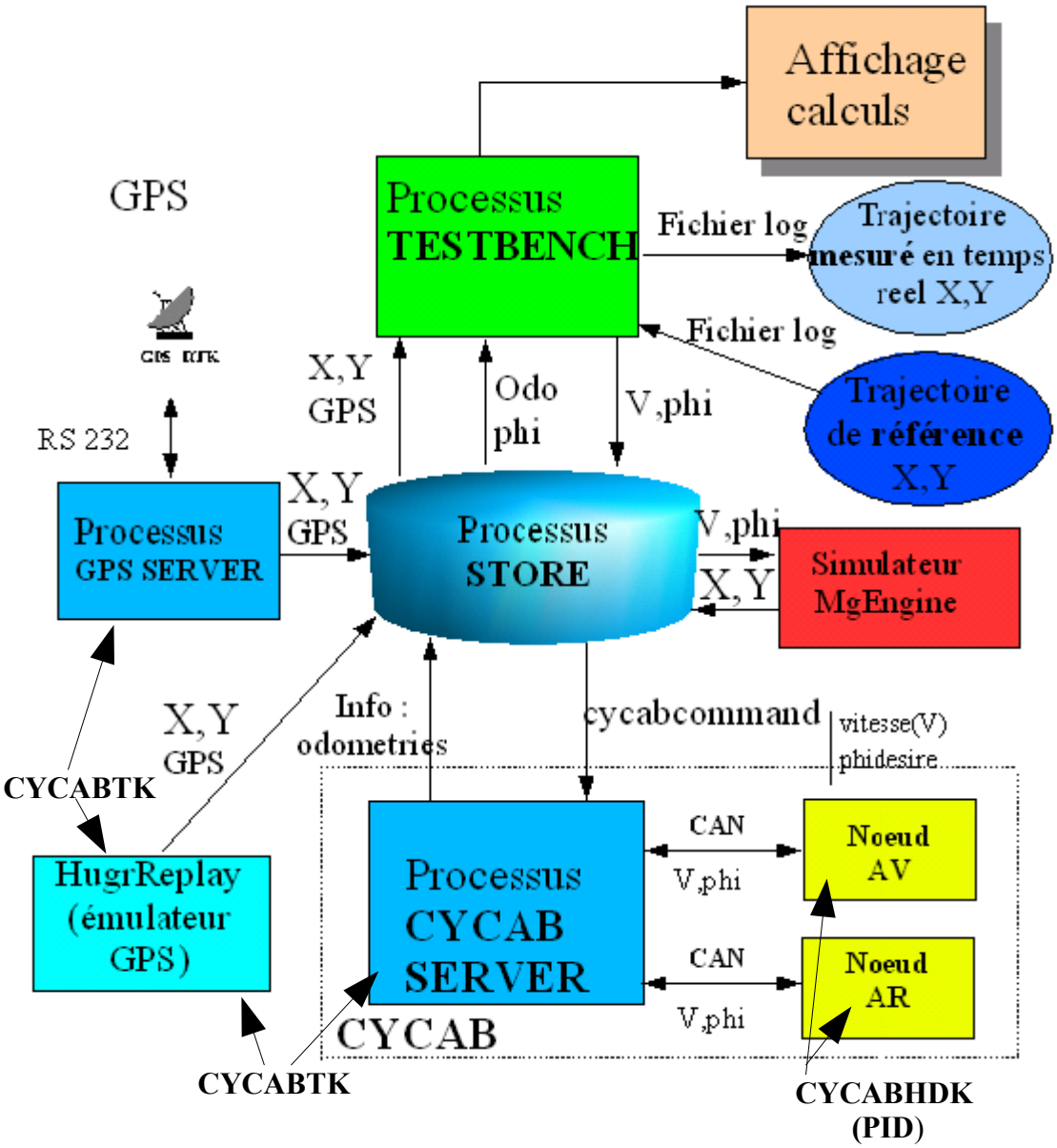
Les deux GPS fonctionnent en même temps. La base compare les coordonnées GPS calculées à partir des satellites avec celles que nous lui avons enregistrées avant la prise de mesure. Une fois l'erreur de positionnement déterminée, il transmet, grâce à son émetteur radio, les données nécessaires pour le GPS mobile. Comme les deux GPS sont assez proches l'un de l'autre comparé à la distance qui les sépare des satellites, l'erreur de positionnement de chaque GPS est quasiment identique. Le GPS mobile en déduit alors sa position réelle en corrigeant sa position calculée avec l'erreur transmise par la base. Grâce à ce principe, ils permettent de s'affranchir des problèmes liées au retard de la troposphère et l'ionosphère et aussi des chemins multiples causes par les ondes

réfléchis.
Une fois l'erreur enlevée, la précision du GPS mobile n'est plus connue au mètre près, mais au centimètre près.
De plus, ces GPS fonctionnent en temps réel, c'est à dire que le calcul de la position est garanti dans une durée de 1/10 seconde, ce qui permet d'avoir de façon certaine une mesure au centimètre près à une fréquence de 10 Hz.

Un synoptique ci dessous permet de bien illustrer le principe de fonctionnement.



4-Architecture logicielle & Implémentation



L` architecture logicielle du cycab est constitué de modules communicants a travers la mémoire partagée sur PC embarqué (LINUX) et un bus CAN pour les contrôleurs bas-niveaux des moteurs . Cet architecture nommée **CYCABTK** & CYCABHDK est développée en interne par le service SED (équipe E-Motion).

4.2.1)CycabTk:

Le Cycab utilise un ensemble de programmes permettant de commander l'ensemble de ses actionneurs en prenant en compte les données reçues par ses capteurs. Ces programmes font partie du dossier **Cycabtk**(Cycab Tools Kit). Il comprend la commande en vitesse, en accélération, en freinage, en direction, mais aussi les programmes d'acquisition des données des capteurs tels que le capteur de détection d'obstacles.
Cet outil est intéressant car il permet d'être intégré directement dans le terminal LINUX embarqué du cycab pour effectuer des tests réels sur le terrain .

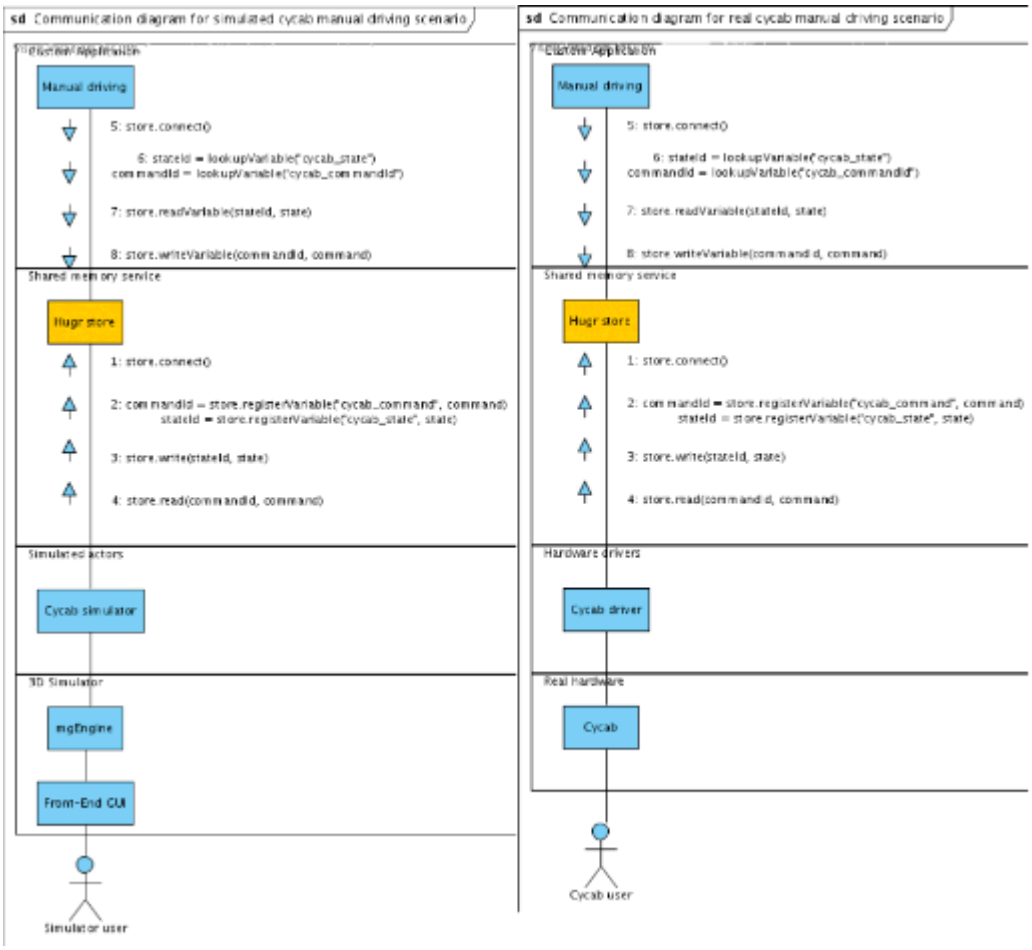
Il est fondé sur un "middleware" appelé **Hugr** permettant un partage de données entre processus sur plusieurs plateformes. Cet intermédiaire permet de passer de la simulation au robot réel sans recompiler.

4.2.2)CycabHDK:

CycabHDK ou (Cycab hardware Kit) est un outil logicielle dédiée à l'interface CAN et l'asservissement PID bas niveaux des nœuds des moteurs .L a compilation croisé de l'API associé aux librairies CAN permettent de générer du code 68000 aux CPU des nœuds des moteurs.

4.2.3)Hugr:

Hugr est une application qui crée des variables dans un système de mémoire partagée afin de les rendre accessibles pour tous les programmes qui en auront besoin. Il a l'avantage de s'affranchir des problèmes temps réel,lorsque plusieurs processus insère des variables dans l'espace de mémoire partagée. **Les 2 diagrammes en UML** permettent de comprendre le fonctionnement (ci dessous).



Definition des principales methodes de Hugr :

1.Déclaration des objets Hugr Store en utilisant les constructeurs **:** **hugr::Store**

2.On récupère un pointeur du store : **hugr::connect()**

3.on enregistre et on alloue les variables dans le Hugr store

```
hugr::registerVariable(variable_name_string, variable_type)
```

4.on récupère l'identificateur de la variable relatif à la variable :_

```
hugr::VariableId hugr::lookupVariable(variable_name_string)
```

5.on lit la valeur de la variable **:hugr::readVariable(hur::VariableId, variable_type)**

6.On écrit la valeur de la variable et la stocke dans le Hugr store :

```
hugr::writeVariable(hugr::VariableId, variable_type)
```

Application qui enregistre une variable de type entier de valeur 42 et la met a jour à 12.

```
#include <iostream>
#include "hugr.hpp"
using namespace std;
using namespace hugr;   // initialisation du package hugr

int main()
{
 try    {
        Store store; //(1)
        int v1 = 42;
        VariableId id1;
        store.connect(); // (2)
        store.registerVariable("variable1", v1); // (3)
        id1 = store.lookupVariable("variable1"); // (4)
        sleep(2); v1 = 12;
        store.writeVariable(id1, v1); // (6)
        }
        catch(const exception &amp;e) { cout << "Writer: " << e.what() << endl;}
        return 0;  }
```

Application qui récupère la variable de type entier provenant du store.

```
#include <iostream>
#include "hugr.hpp"
using namespace std;
using namespace hugr; // initialisation du package hugr

int main()
{
 try     {
         Store store; // (1)
         int v = 0;
         VariableId id;
         store.connect(); //(2)
         sleep(1);
         id = store.lookupVariable("variable1"); // (4)
         store.readVariable(id, v); // (5)
         cout << "variable1 found at id: " << id << " value=" << v << endl;
        }
        catch(const exception &amp;e) {cout << "Reader: " << e.what() << endl;
        } return 0;
```

Hugr est aussi composé de composantes API tel que :

Hugrlog : IL permet de "logguer" des données c'est a dire enregistrer des variables acquises via Hugr dans un fichier.
Hugrreplay: Une fois les données acquises , on peut les rejouer via Hugr
Hugrweb : C'est une API web qui permet d'afficher les variables d'une application sur un navigateur
Un exemple d'application utilisant ces API se trouve en Annexe.

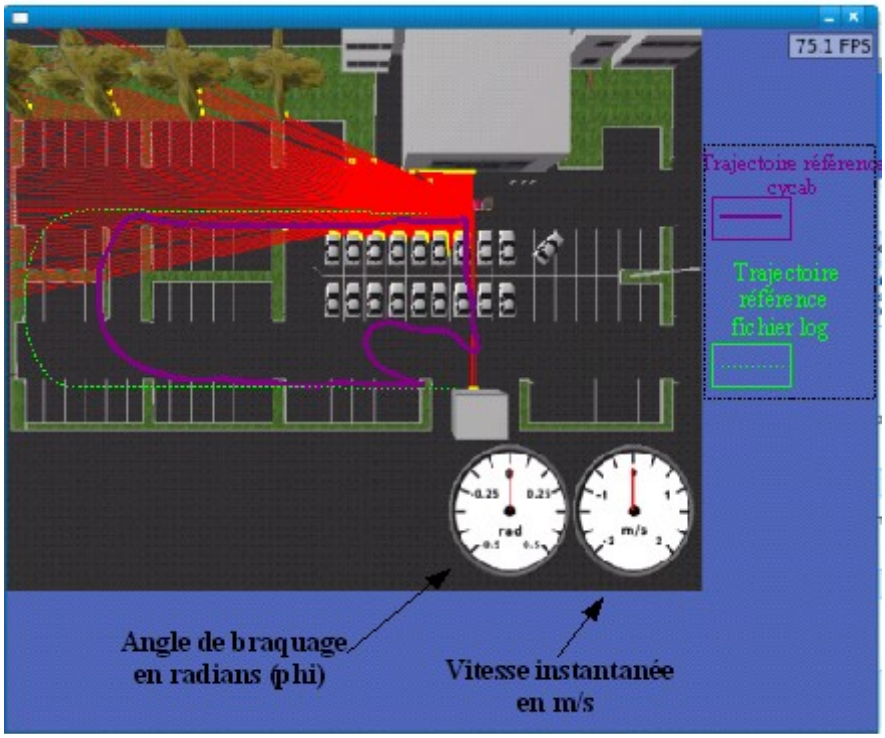
4.2.4)Mgengine

Afin d´ implémenter les tests , les modules commandant des actionneurs et des capteurs sont remplacés par des simulateurs ayant la même interface logicielle.
Le moteur de rendu et de simulation utilisé est **Mgengine** , son originalité permet de rajouter très rapidement et facilement des capteurs et des robots.

Ces capteurs peuvent émuler soit des cameras ,télémètres laser ou GPS.En fournissant 2 variables de commandes, l'angle et la vitesse ,**Mgengine** simule le comportement dynamique d'un véhicule (vitesse, position,accélération) mais uniquement dans les conditions idéales de roulement sans glissement.

Avant de lancer le simulateur , il est important d'initialiser la mémoire partagée comme suit :

1. **hugrstore &**
2. **mgengine**



4.2.4-Simulateur Mgengine

5-ETAT DE L'ART

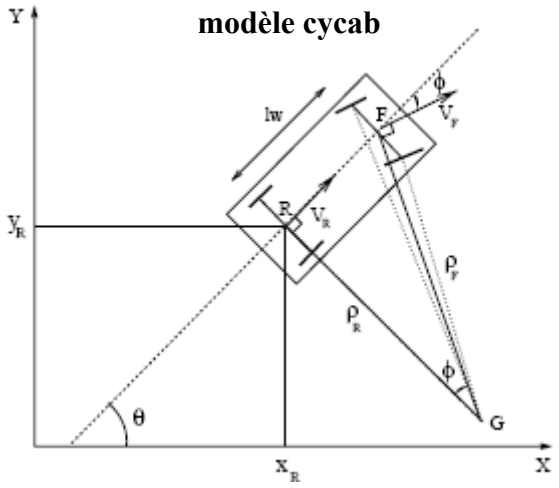
5.1) travail de préparation :

.

Ma remise a niveau en **Automatique avancée** et mes recherches bibliographiques (IEEE, ressources INRIA) m'ont permis de mieux cerner le projet du stage.
Pour bien comprendre le modèle du cycab ,il faut au préalable étudier sa cinématique .

5.1.1)Cinematique du cycab[1]:

modèle cycab

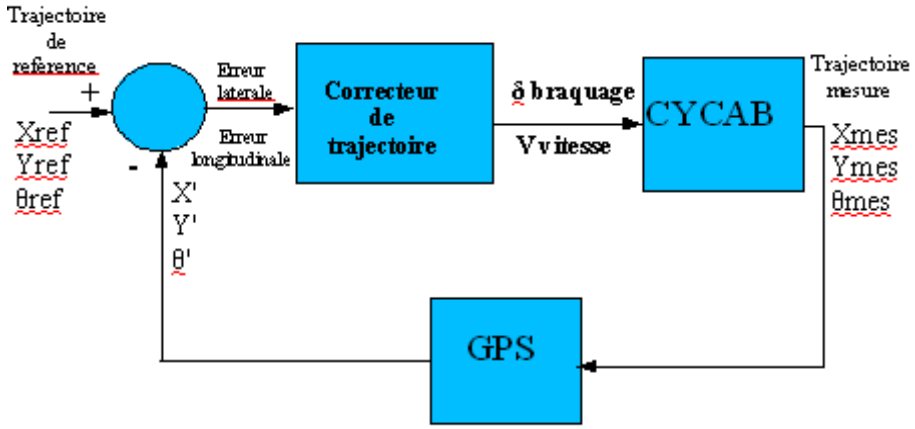


La figure ci contre permet de comprendre la modélisation du cycab avec ses 2 variables de commandes (Vitesse(VF),angle phi(Φ)).

Soient F = (xF ; yF) (resp. R = (xR; yR)) les coordonnées du centre de l'essieu avant (resp. arriere). De même, **vF** (resp. vR) représente la vitesse instantanée au point F (resp. R).
Φ représente le braquage moyen des roues de l'essieu avant.
lw correspond a l'empattement du véhicule.
θ caracterise l'angle que fait l'axe longitudinal du vehicule avec l'axe des abscisses du repère lie a l'environnement.Enfin,**ρF** (resp. ρ R) est le rayon de giration instantanée associé au point F(resp.**ρ**R)

Le synoptique ci dessous permet de bien schématiser le système en boucle fermée autour du cycab.

5.1.2)synoptique du projet :



Les schémas blocs du synoptique sont :

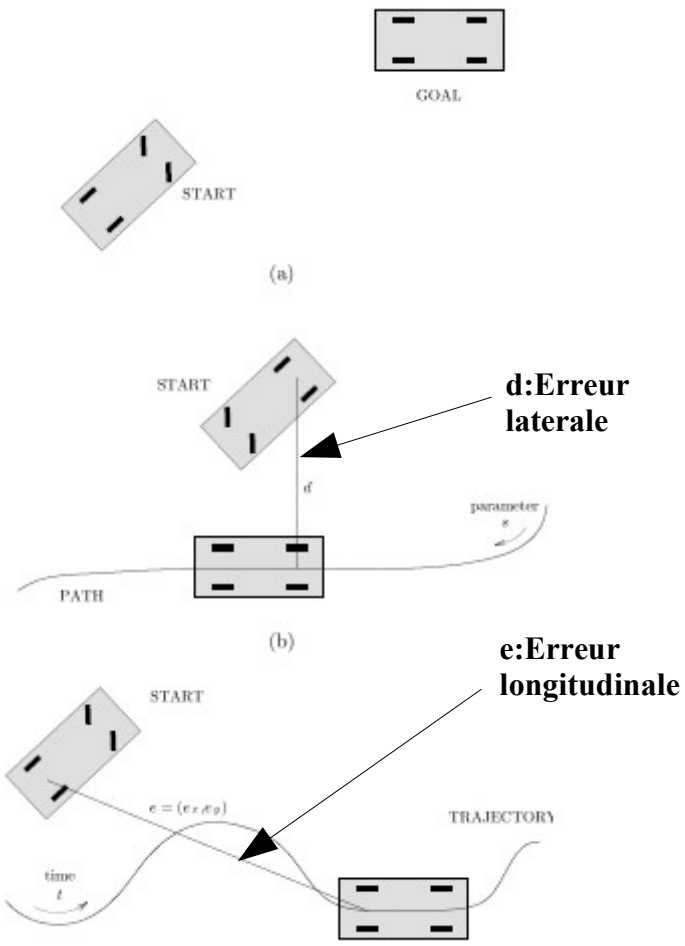
- le correcteur de trajectoire ou calculateur avec son implémentation purement logicielle.
- le procédé cycab .Sa constante du temps est estimé à 10 ms car elle prend en compte l'asservissement PID bas-niveaux des moteurs .
- Le GPS qui échantillonne les positions toutes les 100 ms soit à une fréquence de 10 hz.

Afin d'éviter d'avoir des problèmes de temps réel au niveau du système , l'algorithme du correcteur doit s'assurer que le traitement des données et le calcul de la loi de commande doivent s'exécuter dans un temps imparti et inférieure à 10 ms.

5.2-Classification des taches du cycab

D'après [2],on distingue 3 types de mouvements :

-mouvement point a point :Le cycab doit atteindre une position désirée a partir d'une configuration initiale (a)
-le suivi de chemin (path following):Le cycab doit atteindre et suivre une trajectoire dans une représentation cartésienne a partir d'une configuration initiale. La vitesse n'est pas commandée et ne varie pendant le parcours (V=cste)sauf en cas de contraintes extérieures.(b)
-la poursuite de trajectoire(trajectory tracking) : le cycab doit atteindre et suivre une "trajectoire mouvante"qui suit une loi paramétrique du temps t .L'erreur longitudinale doit être prise en compte.



choix adopté:

A travers ce stage , nous mettrons en œuvre une loi de commande avec **suivi de chemin** et dont seul **le paramètre 's'** ou abscisse curviligne interviendra .
En effet , elle permet plus de souplesse et de simplicité dans la loi de commande car elle est indépendante de la vitesse linéaire et sans contrainte de temps.
De plus, on peut contrôler la vitesse en cas de contraintes extérieures :

- évitement d'obstacle
- mode manuel pour l'aide à la conduite (retour à un point précis ou lors de virages très courbé)

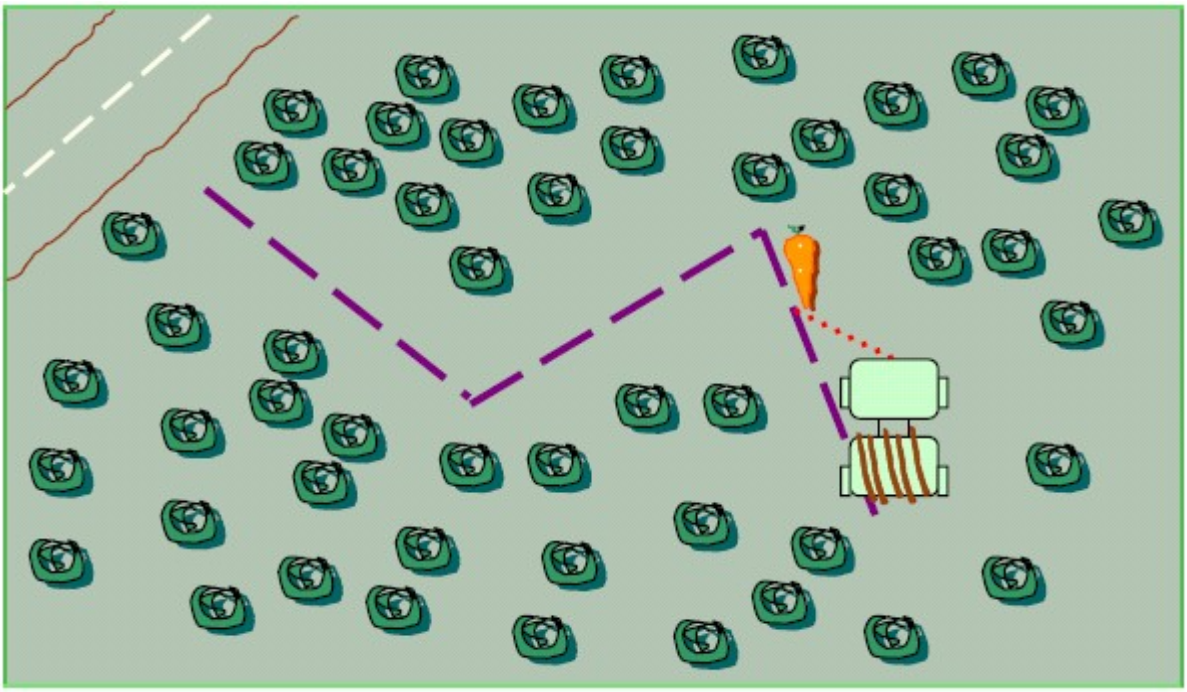
Dans ce cas , seule **l'erreur latérale** sera considérée dans la loi de commande.

5.3-Suivi de chemin

Nous allons présenter 3 approches de suivi de chemin : «follow the carrot","pure pursuit ",vector pursuit" et les systèmes contrôles "chainées ".

5.3.1)Follow the carrot

Cette appellation “FOLLOW THE CARROT” [3] vient du fait que pour définir un chemin il fallait fixer une carotte devant un animal pour l 'appâter tout en le faisant suivre une trajectoire donné .Le “point de carotte” est définie comme le point du chemin a une certaine distance de la cible.
L'erreur d'orientation du véhicule est définie comme l'angle courant du véhicule et la ligne en pointillée au 'point de carotte'.Un erreur d'orientation nulle signifie que le véhicule se trouve au point de carotte.
Cette approche a ses inconvénients car le véhicule a tendance a osciller pendant les virages.



(Figure 5.3.1) :Poursuite de trajectoire définie par des points de navigation

5.3.2)Pure poursuit

cette algorithme[4] calcule la courbure de l'arc pour atteindre un objectif désiré.

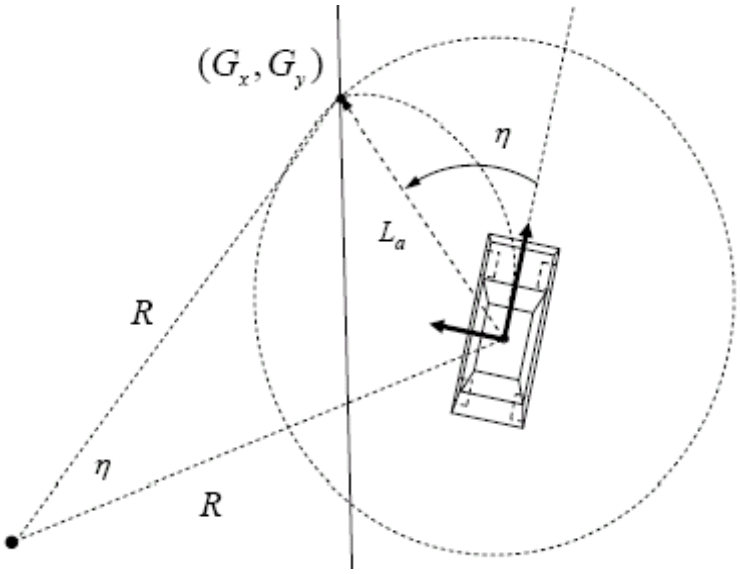
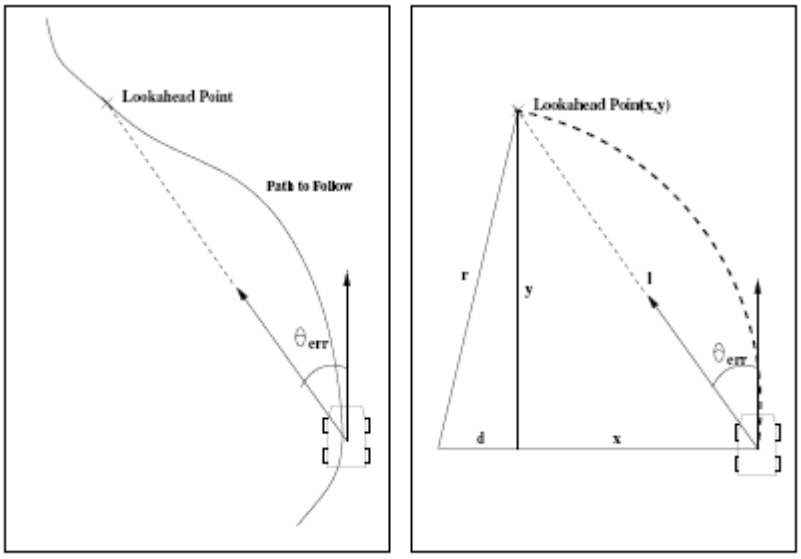


Figure 5.3.2 : pure poursuit par point d'objectif

Comme le montre la figure 5.3.2 , cette approche utilise un paramètre réglable "*La"(lookahead)* qui définie un cercle virtuelle autour du véhicule . L'intersection du cercle avec le trajectoire spécifié définie un point objectif G(Gx,Gy) .
L'algorithme de pur poursuite spécifie soit un courbure désirée ou une accélération centripète désirée basée sur l'arc du chemin. Cet arc présente 3 contraintes :

- Il doit être tangent au vecteur vitesse situé à l'origine du repère fixe du véhicule.
- I doit traverser l'origine du repère et le point objectif .
- Le rayon de courbure doit coïncider avec le rayon de giration instantané .



La courbure nécessaire pour atteindre l 'objectif est donné par la formule :
κ=sin(η)/La
et la la loi de commande pour l'angle de braquage est :
δ=tan $^{-1}$****2*Lb*sin(η)/La*** avec **Lb**: l'empattement du véhicule cad la distance entre l'essieu avant et arriere

on constate qu'elle est non linéaire et fonction de la courbure ***η***

5.3.3)vector poursuit

Cette algorithme [4] est différente de la précédente car elle considère non seulement le point de carotte mais également la possible orientation du véhicule au point de carotte. Cela signifie qu'il arrive au point de carotte avec une orientation et un angle de braquage correcte . Pour cela elle utilise la " théorie des torseurs ".

comparatifs :

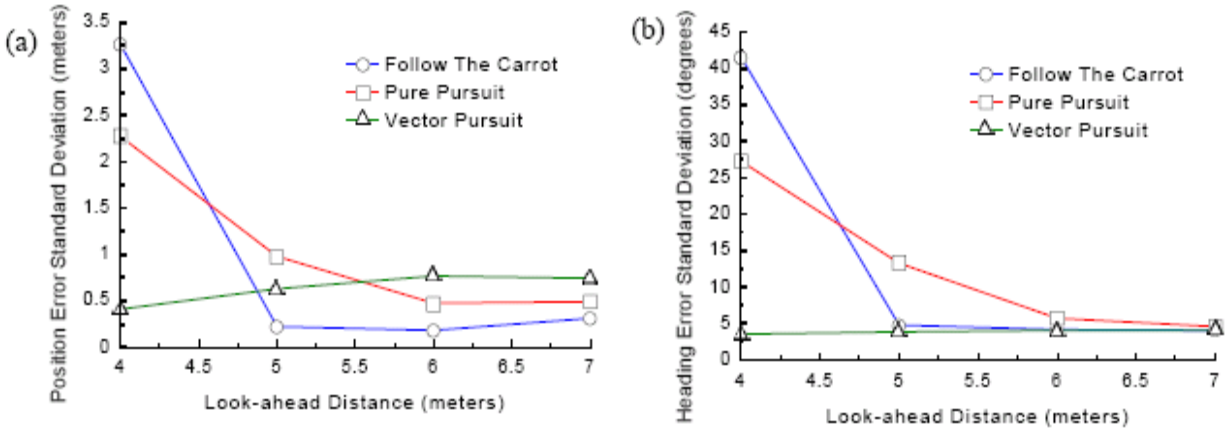


figure 5.3.3 :erreur de position latérale et angulaire en fonction de *La(lookahead)*

Choix de l'algorithme :

Le suivi de chemin de véhicules tel que le cycab' a fait l'objet d'une collaboration entre le LASMEA (laboratoire de recherche à Clermont-Ferrand) et l'INRIA.
En effet le LASMEA a pu tester sans difficulté sa loi de commande sur des machines agricoles (tracteur) puis porter sur des véhicules "cycab".
Le LASMEA [6] utilise pour cela , un contrôle de systèmes chaînées qui permettent de linéariser exactement et sans approximation (sin (x)≃x) le modèle non-linéaire du cycab .

Afin de faciliter l'implémentation du projet ,le choix de l'algorithme s'est donc orienté sur les systèmes chaînées .

Faute de temps,les algorithmes précédents (follow the carrott , pure poursuit,vector pursuit) n'ont pas été implémentés .

5.4-Modélisation du cycab

Dans un premier temps le schéma classique type voiture sous hypothèse de roulement sans glissement(modèle Ackerman) est représente sur la figure 1.Le véhicule est considéré comme étant un modèle bicycle(1 roue considérée comme le train avant et une pour le train arrière).

- C est la trajectoire à suivre .
- O est le centre de l'essieu arriéré,
- M est le point de C le plus proche de O
- M est considéré comme unique , ce qui est vrai lorsque O est suffisamment proche de C
- s est l'abscisse curviligne de M le long de C , et c(s) représente la courbure de C en ce point.
- Y et $\tilde{\theta}$ sont respectivement l'écart latéral et angulaire du véhicule par rapport a la trajectoire de référence C . θc étant le cap du véhicule (voir figure 1)
- δ est l'angle de braquage virtuel de la roue avant .
- υ est la vitesse linéaire du véhicule , considéré ici comme un paramètre , dont la valeur peut varier pendant le guidage.
- L est l'empattement du véhicule

En utilisant cette modélisation , il est possible de définir les équations du mouvement du véhicule considéré comme un bicycle, sous l'hypothèse du roulement sans glissement, donne par l'équation 4.1.Un tel modèle permet de caractériser le mouvement du véhicule tant que l'hypothèse d'application est vérifiée.

$$\begin{aligned}\dot{s} &= \frac{v\cos(\tilde{\theta})}{1-c(s)y} \\ \dot{y} &= v\sin(\tilde{\theta}) \\ \dot{\tilde{\theta}} &= v\left(\frac{\tan(\delta)}{L} - \frac{c(s)\cos(\tilde{\theta})}{1-c(s)y}\right)\end{aligned} \qquad \textbf{(4.1)}$$

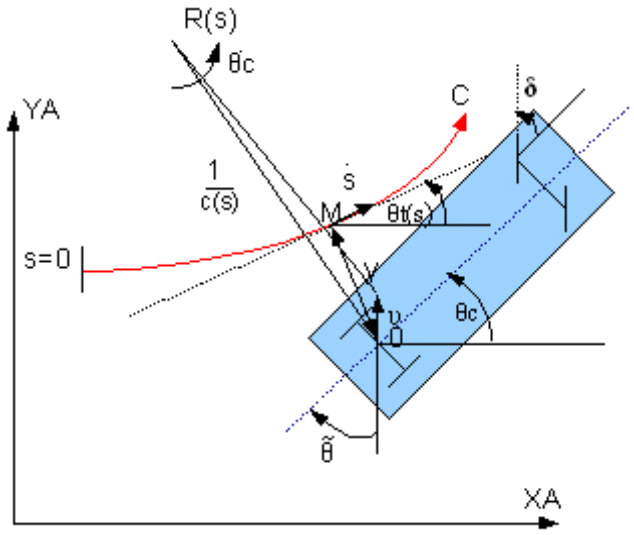


figure 1 – Modèle cinématique du véhicule cycab

L' existence de ce modèle est assurée sous la condition suivante :

$1 - c(s)y \neq 0 \quad \Rightarrow y \neq 1/c(s)$ **(4.2)**

D'un point de vue physique , la condition signifie que le point 0 ne se situe pas au centre de la courbure de la trajectoire C (en d'autres termes , le rayon de courbure de la trajectoire doit être différent de l'écart latéral).En pratique , cette condition est toujours satisfaite, compte tenu des écarts relativement modestes par rapport aux autres ordres de grandeur du rayon de courbure

5.5-Linéarisation exacte -contrôle de systèmes chaînées

Le modèle (4.1) d'évolution du cycab non holonome est non linéaire (ni en l'état, ni en la commande).Cependant , il a été prouvé dans ([5], voir bibliographie) qu'un tel système pouvait être converti par des changements de variables en un système dit «chainé», beaucoup plus simple à manipuler en vue de la commande. Ainsi, les transformations d'états et de commande données par (4.3) permettent de mettre le système (4.1) sous la forme (4.4).

$$\Theta((s, y, \tilde{\theta})^T) = (s, y, (1 - c(s)\, y)\tan(\tilde{\theta}))^T = (a1, a2, a3)^T$$

$$M(\upsilon, \delta) = \left(\frac{\upsilon \cos(\tilde{\theta})}{1 - c(s)\, y}, \frac{d}{dt}(\tan(\tilde{\theta})[1 - c(s)\, y])\right)^T = (m1, m2)^T \qquad \textbf{(4.3)}$$

Les transformations Θ et M sont inversibles sous les conditions 1-c(s)y ≠0, υ ≠0, ce qui a déjà été admis compte tenu des hypothèses de travail et sous la condition $\tilde{\theta} \neq \pi/2 \bmod \pi$. Dans la pratique , on pourra en effet considérer que l'écart angulaire restera dans l'intervalle]- π/2; π/2[.

$$\begin{cases} \dot{a1} = m1 \\ \dot{a2} = a3\, m1 \\ \dot{a3} = m2 \end{cases} \qquad \textbf{(4.4)}$$

le modèle (4.4) constitue un système chainé, dépendant de 2 variables de commande :m1, qui est homogène a la vitesse d'avance ($\dot{s}$)du véhicule le long de C et m2, dépendant de l'angle de braquage (δ).qui est la seule variable commandée dans notre cas.la dépendance de m2 à l'angle de braquage intervient par l'intermédiaire de la dérivée de l'écart angulaire par rapport au temps ($\dot{\tilde{\theta}}$)dont une expression la liant à δ est donnée par (4.1).
Afin de construire une loi de commande dont les performances seront indépendantes de la vitesse υ d'avance du véhicule , il est judicieux de récrire le système chaîné (4.4) par rapport à l'abscisse curviligne plutôt que par rapport au temps. En notant an'=d (an)/ds, le système devient :

$$\begin{cases} a1' = 1 \\ a2' = a3 \\ a3' = \dfrac{m2}{m1} = m3 \end{cases} \qquad \textbf{(4.5)}$$

la conversion du système (4.1) en un système chainé (4.4), puis le changement d'échelle des temps permet donc d'obtenir une forme linéaire exacte (4.5).Ceci permet d'appliquer de techniques de commandes performantes issues de l'automatique linéaire , bien maitrisées.

5.6-Loi de commande en l'absence de glissement

en utilisant le système linéarisé exacte (4.5), une loi de commande dédiée au suivi de la trajectoire *C* peut être calculée de la façon suivante ,l'objectif de suivi de trajectoire est d'asservir les écarts latéral ta angulaire par rapport a la trajectoire de référence à 0.
Dans le cas du système (4.5) , cet objectif se traduit par la convergence des variables *a2* et *a3* à 0 .
Cette convergence peut être obtenue par le choix d'un régulateur Proportionnel Dérivé pour la variable de commande m3, comme suit :

$m3=-Kda3 - Kpa2 \quad (Kp,Kd) \in \mathbb{R}_{+2}$ **(4.6)**

ou Kp et Kd sont deux réels définis positifs fixant la réponse du système, homogènes à des gains proportionnel et dérivé d'un correcteur classique. Un tel choix sur *m3* permet effectivement d'aboutir à l'équation différentielle suivante :

$a2'' + Kda2' + Kpa2 = O$ **(4.7)**
en résolvant cet équation du 2nd degré , les paramètres doivent satisfaire la relation : **Kd=2√Kp.**

ce qui implique la convergence souhaitée .
Le système étant définie par rapport a l'abscisse curviligne et non par rapport au temps .Les 2 gains Kd et Kp fixent non pas un temps de réponse mais une distance de réponse théorique , indépendante de la vitesse longitudinale du véhicule. En utilisant l'expression de la commande (4.6) ainsi que le changement de variables (4.3) , on obtient finalement l'expression non linéaire (4.8) de la loi de commande à appliquer sur l'actionneur permettant le braquage des roues avants.

$$\delta_{RSG}(y,\tilde{\theta})=\arctan\left(L\left[\frac{\cos^3(\tilde{\theta})}{(1-c(s)y)_2}\left(\frac{dc(s)}{ds}y(\tan(\tilde{\theta})-Kd(1-c(s)y)\tan(\tilde{\theta})-Kpy+c(s)(1-c(s)y))\tan^2(\tilde{\theta})\right)+\frac{c(s)\cos(\tilde{\theta})}{1-c(s)y}\right]\right)$$

(4.8)

L'expression **(4.8)** est bien entendu définis sous les 3 conditions précédemment évoquées ($y\neq0$, $\upsilon \neq 0$, et $\tilde{\theta} \neq \pi/2 \bmod \pi$.

On peut constater que la formule (4.8) peut être simplifiée pour des trajectoires a courbure nulle c'est a dire c(s)= 0

$\delta(y, \tilde{\theta}) = \arctan(l\cos^3 \tilde{\theta}(-k\tan \tilde{\theta} - Kpy)$ **(4.9)**

Traitement des données du capteur GPS :

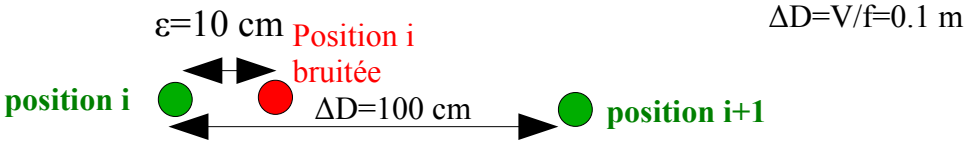
Les bruits apportés au capteur GPS peuvent provoquer des erreurs de positionnement.
Afin de ne pas introduire de bruits dans la loi de commande , il est important de traiter et de filtrer les données .
Le traitement des signaux GPS sera réalisé au moyen d'un filtre/estimateur KALMAN.

6-Filtrage et Estimation KALMAN des données GPS

Le filtre KALMAN a l'avantage de minimiser la variance de l'erreur d'estimation (voir ANNEXES)

Pour tester le filtre , il faut faire une acquisition d'une trajectoire réelle en conduisant manuellement le cycab à l'aide du joystick .

Condition d'essais : Vitesse d'avance et constante du cycab : V= 1 m/s;
données GPS : f= 10 Hz; précision : 1 cm .

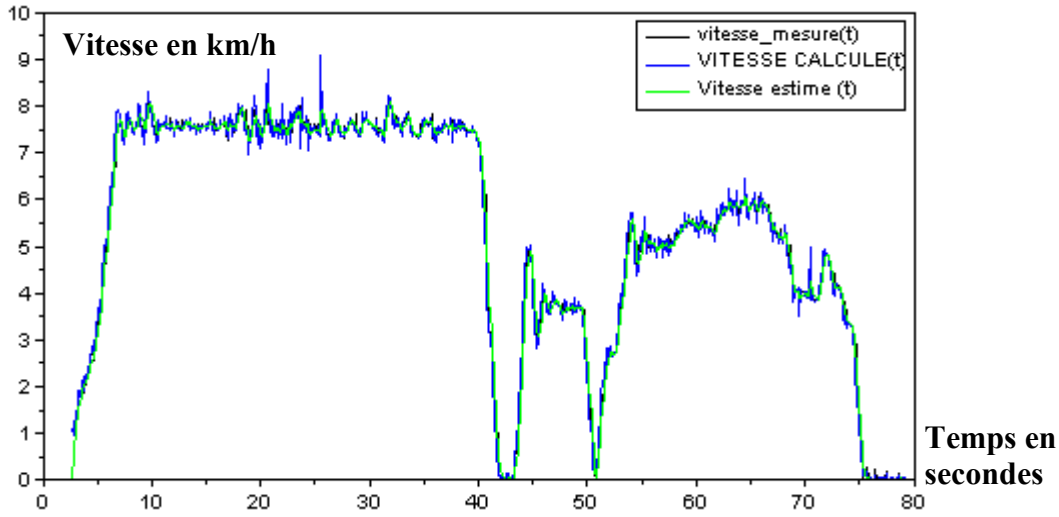


La figure ci dessus permet de montrer l'espacement des échantillons des positions GPS avec les conditions précédentes .Si le GPS présentait une erreur de positionnement de 10 cm , cela signifie qu'entre 2 échantillons l'erreur relative est de $\varepsilon r=\varepsilon/\Delta D= \varepsilon.f/V$ soit 10% ce qui est considérable et inacceptable pour corriger la trajectoire .On en deduit que l'erreur relative est inversement proportionnelle à la vitesse d'avance du cycab .Le problème ne se poserait pas si la trajectoire est synthétisé par un fichier car les positions ne sont pas très espacés.La seule contrainte est de programmer une courbe lisse .
Pour éliminer ces bruits, le filtre KALMAN a été simulé sous SCILAB .

Simulation sous SCILAB
la simulation sous SCILAB a permis de filtrer une trajectoire avec un signal très bruité.
Les principaux paramètres de réglage sont :

- Q pour la matrice de covariance bruit du modèle avec comme valeur Qv=0.05
- R pour la matrice de covariance bruit de mesure avec comme valeur Rv=0.01

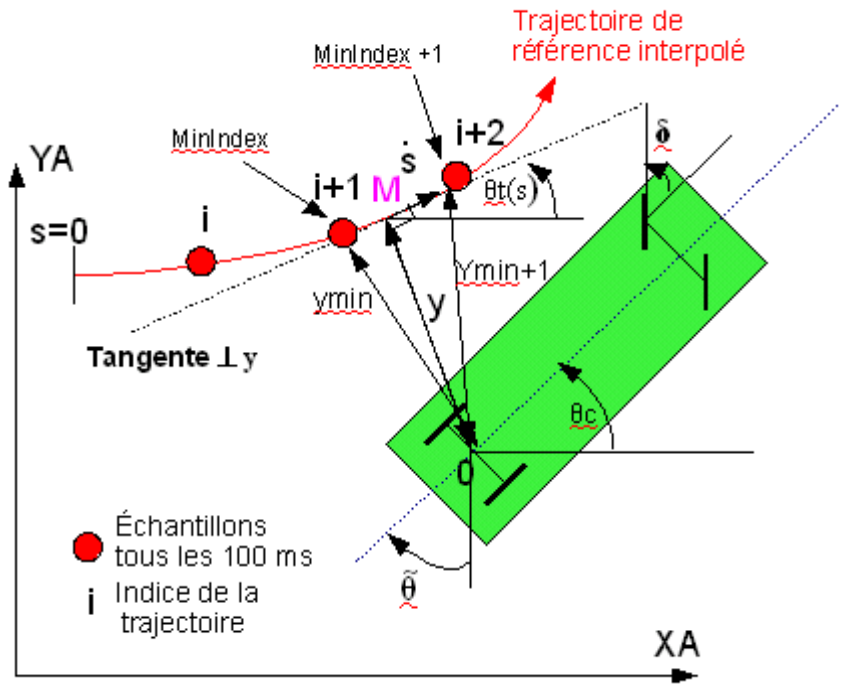
La simulation ci dessous a permis de bien mettre en évidence le rôle du filtre KALMAN
En choisissant **Q=[0 0 0 0 ;0 Qv 0 0;0 0 0 0;0 0 0 Qv]; R=[Rv Rv;Rv Rv]** , le signal (en vert) est bien filtré et suit correctement le signal .On a pu constaté qu`en augmentant les valeurs de R c'est à dire en passant de 0.01 a 0.5 , on lissait davantage la courbe mais on introduisait du retard



Implémentation de l'algorithme sous C++:L'algorithme a été implémenté sous C++ a l'aide des librairies BOOST qui intègre les routines matricielles indispensable du filtre KALMAN .

7-Interpolation de la trajectoire de référence

L'interpolation serait inutile si la trajectoire de référence aurait été synthétisé par un fichier.
Sachant que le GPS échantillonne les positions tous les 100 ms , il est important d'interpoler entre les valeurs intermédiaires repérées par les indices i+1 et i+2.
En effet , le programme calculant le point le plus proche repéré peut soit donner une déviation latérale égale à ymin ou ymin+1 alors que la déviation idéale est y.
Par conséquent , on commet des erreurs non négligeable pour la déviation latérale ,l'angle θt de la trajectoire de référence et donc la commande δ .



Rappel :
La loi de commande avec courbure (équation **4.8**) est :

$$\delta_{RSG}(y,\tilde{\theta}) = \arctan\left(L\left[\frac{\cos^3(\tilde{\theta})}{(1-c(s)y)^2}\left(\frac{dc(s)}{ds}y(\tan(\tilde{\theta}) - Kd(1-c(s)y)\tan(\tilde{\theta}) - Kpy + c(s)(1-c(s)y))\tan^2(\tilde{\theta})\right) + \frac{c(s)\cos(\tilde{\theta})}{1-c(s)y}\right]\right)$$

la loi de commande sans courbure (équation **4.9**) est :

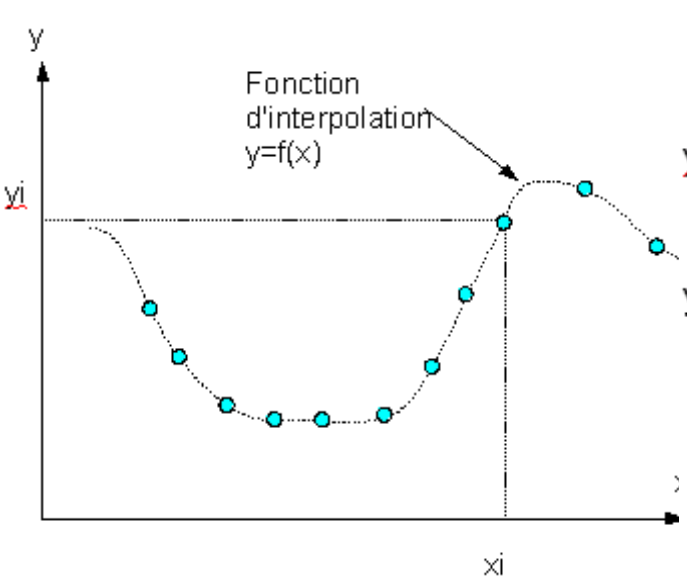
$\delta(y, \tilde{\theta}) = \arctan(l\cos^3\tilde{\theta}(-kd\tan\tilde{\theta} - Kpy)$

L'interpolation linéaire est primordiale pour avoir une bonne précision de la déviation y et aussi de la commande .
Concernant la loi de commande avec prise en compte de la courbure , l'interpolation polynomiale ou spline(voir annexes) peut aussi être implémenté pour une détermination exacte de la courbure **c(s)** et de sa dérivée **ds** .

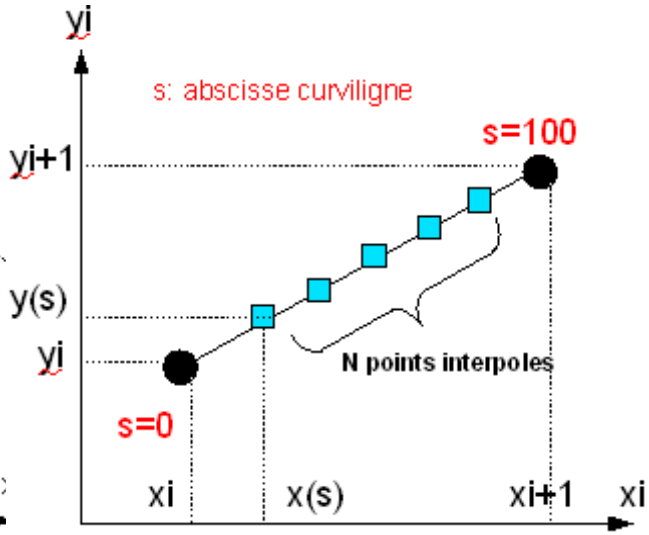
7.1)Méthode d'interpolation:

Rappel:

On connait N+ 1 points ou échantillons de la trajectoire de référence , l'opération d'interpolation consiste à estimer la valeur d'un point d'abscisse x intermédiaire aux "xi".Les points (xi,yi)sont appelés point d'interpolation. L'estimation de y est réalisée a l'aide d'une fonction d'interpolation qui passe par ces points .(figure ci-dessous)



Fonction d'interpolation d'une la trajectoire de référence (GPS) échantillonné tous les 100 ms



interpolation linéaire

7.2)Interpolation linéaire

Dans ce cas , les points intermédiaires calculés sont placés sur la droite passant par les 2 échantillons :
Pour retrouver les positions x(s) et y(s) , il faut raisonner en représentation paramétrique c'est a dire en fonction de **'s** 'ou abscisse curviligne avec **s=[0..100]** ou en fonction du temps **'t' et t=[0..Te].** ou Te= Période d'échantillonnage .

Les équations paramétriques sont :

$$\begin{cases} x(s) = xi + \dfrac{(x(i+1) - x(i))}{100} * s \\ y(s) = yi + \dfrac{(y(i+1) - y(i))}{100} * s \end{cases}$$

Expérimentation:

Pour les essais , l'interpolation linéaire a été utilisé avec un N= 100 .Le cycab se déplaçait à une vitesse constante de 2 m/s et sachant que les échantillons de positions GPS sont relevés à 10 Hz(Te=100 ms) , les points de la trajectoire de référence sont espacés de 20 cm .
Sans interpolation , la déviation latérale oscillait entre 2 extrêmes soit **+/- 0.3 m** et engendrait un fort bruit dans la commande.La présence de l'interpolation linéaire a permis de faire converger l'erreur latérale en dessous du **centimètre** ce qui correspond à la précision du capteur GPS .

7.3)Polynôme d'interpolation :

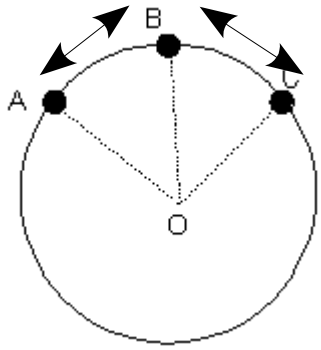
Rappel : par 2 points passe une droite , par trois points passe une parabole donc N+1 points différents passe un unique polynôme de degré N
Le polynôme est exprimé par $P(x)=a0+a1x+a2x^2+...+aNx^N$

Plusieurs méthodes d'interpolation tel que la méthode de Lagrange ou de Newton ont été implémentés dans le programme .
L' interpolation spline est une forme d'interpolation ou l'interpolant est une fonction polynomial par morceaux qu'on appelle spline. Elle est plus intéressante que les précédents car elle garantie une erreur d'interpolation plus faible.

Faute de temps,l' interpolation spline a été uniquement testé en simulation avec MgEngine afin d' implémenter la loi de commande avec courbure .

7.4)détermination de la courbure

7.4.1)Methode par le cercle circonscrit



On peut faire passer un cercle par 3 points :L'algorithme énoncé en annexes , permet de retrouver le centre du cercle passant par ces 3 points et de déterminer la courbure sur une plage de points.

Simulation: En implémentant cette algorithme sur une trajectoire de référence réelle , on a pu obtenir une courbure qui était assez fidèle à la trajectoire. En réalité ,la trajectoire de référence étant bruitée et non régulière, entrainait des erreurs au niveau de la courbure .L' avantage de cette technique est qu'on peut moduler l'espacement des points (20 points) et de permettre une meilleur prédiction de la courbure et anticipation de la commande.

7.4.2)Methode par les équations paramétriques :

A partir des échantillons de la trajectoire de référence , on peut créer 2 fonctions bidimensionnelles cubique spline en prenant 3 points de contrôles consécutives .
Les 2 fonctions paramétriques polynomiale et cubique par morceau pour chaque segment et pour chaque coordonnes X et Y (positions GPS) sont :

$$\begin{cases} \mathbf{X(S)= Ax.S^3 + Bx.S^2 + Cx.S^2 + Dx} \\ \mathbf{Y(S)= Ay.S^3 + By.S^2 + Cy.S^2 + Dy} \end{cases}$$ le paramètre S étant l'abscisse curviligne.

La courbure est donc exprimé par :

$$\kappa(s)=\frac{\dot{x}(s)\cdot\ddot{y}(s)-\dot{y}(s)\cdot\ddot{x}(s)}{(\dot{x}(s)^2+\dot{y}(s)^2)^{3/2}}$$

expérimentation:
L'implémentation de cette technique a été simulé avec MATLAB (**voir ANNEXES**).On obtient des résultats assez similaire de la technique précédente .2O points d'espacement ont aussi été nécessaire afin d' éliminer les bruits qui ne rendait pas la trajectoire lisse et uniforme.

8-Conditions d'expérimentations du cycab

Les essais se sont déroulés fin juillet (beau temps) en même temps que la simulation de la loi de commande avec courbure .
Pour réaliser les différents essais du cycab ,il faut au préalable remplir plusieurs conditions :

8.1)la réservation du parking :Il faut réserver des places de parking pour une durée limitée (2 semaines) .

8.2)Sécurité du cycab :
Afin de na pas générer des surcharges au niveau des moteurs , il est important de prévoir une rampe d'accélération et de décélération pour le démarrage et l'arrêt du cycab .
De plus , un arrêt d'urgence a été intégré dans le cycab en cas d'incident de parcours .

8.3)Sécurité du GPS :

Pour recevoir les données GPS , le programme lit un champ nommé «mode de fonctionnement »
Si ce champ est égal à 5 , le GPS fournit des données incohérentes et erronées car il perd des satellites pour sa localisation.
Seulement 4 satellites sont nécessaires pour avoir un positionnement tri dimensionnelle et la vitesse d'un récepteur.
Tous les essais qui vont suivre ont été réalises avec la loi de commande ne prenant pas en compte la courbure de la trajectoire (équation 4.9).
$\delta(y, \tilde{\theta}) = \arctan(l\cos^3\tilde{\theta}(-k d \tan\tilde{\theta} - Kpy)$ avec **y** représentant la déviation latérale , **δ** la commande ou l'angle de braquage à appliquer aux roues avants.
La vitesse **V** sera constante tout le long du parcours mais peut être aussi utilisé comme un paramètre de la loi de commande .

Pour valider cette loi de commande sur le terrain, nous ferons varier plusieurs paramètres :

- **Kp** ou gain proportionnel
- **Kd** ou gain dérivé
- **V** ou la vitesse d'avance du cycab
- La présence ou non du filtre Kalman pour la trajectoire de Référence. En effet , ceci est très utile si nous souhaitons réaliser une poursuite de trajectoire(trajectory tracking) d'une cible mouvante.
- Le filtrage en temps réel des positions GPS du cycab.
- **L'interpolation linéaire** des échantillons de la trajectoire de référence .

8.4)Prise en main du simulateur et test sur une trajectoire synthétisé par un fichier

Nous avons créer un fichier trajectoire synthétique composés de portions rectilignes et demi-circulaire à *vitesse constante* (4.3.4 p 11).En comparant la trajectoire de référence et celle mesuré par le simulateur , nous avons pu constaté que la correction de trajectoire fonctionne très bien .

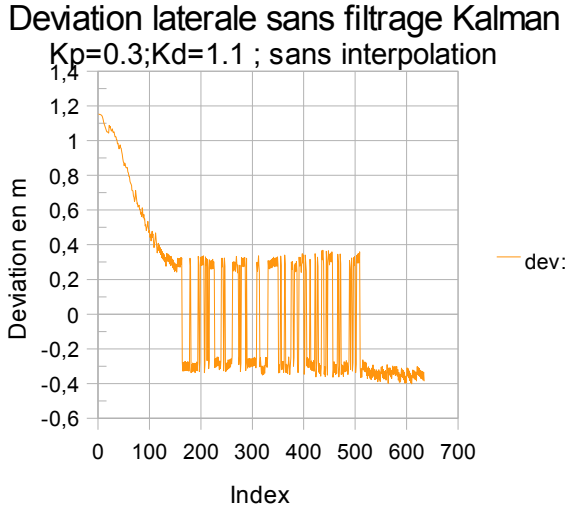
8.5)Test sur une trajectoire réel du cycab (déplacement au parking de l'INRIA).
En pilotant manuellement le cycab avec le joystick , le GPS enregistre les données (Nord,Est,cap, position, vitesse) de la trajectoire réel (cf figure 4.2.4 simulateur MgEngine).Afin d´exploiter et de mieux traiter ces données , nous utilisons le logiciel SCILAB.
Cet outil m`a aussi permis d´implémenter un filtre /estimateur KALMAN pour filtrer les bruits qui sont à l´origine des accélérations brusques du cycab .

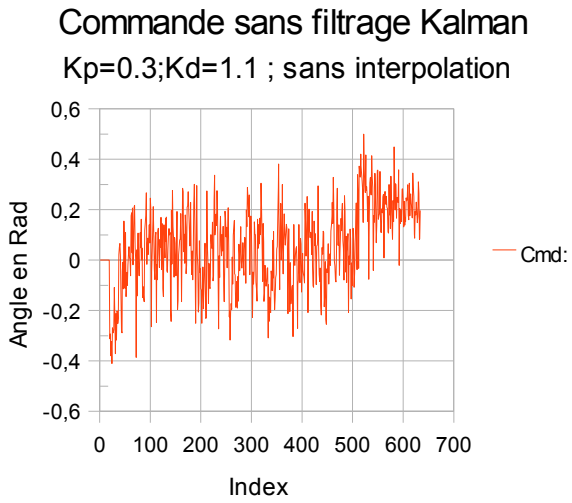
9-Relevés et exploitation des résultats:loi sans courbure

Conditions d'essais:

En conduite manuelle,le cycab enregistre une trajectoire à une vitesse de 2 m/s.
En mode automatique , les essais s'effectueront à faible vitesse c'est a dire à 0.5 m/s ;1 m/s et 2 m/s.

9.1.1)Deviation et commande sans interpolation linéaire et sans filtrage Kalman





Index : ce paramètre représente la position des points

Lorsque la trajectoire de référence n'est ni filtré ou interpolé , la déviation latérale est beaucoup plus bruitée et oscille entre 2 valeurs extrêmes soit 0.3 et -0.3.En effet ,ces fluctuations sont dues d'après l'algorithme au calcul du point le plus proche qui est soit tantôt l'échantillon avant ou arrière de la trajectoire.
Sachant que cet essai a été réalisé à une vitesse de 2 m/s et que les échantillons GPS sont à 10 Hz on retrouve bien les points de la trajectoire de référence espacés à 20 cm.

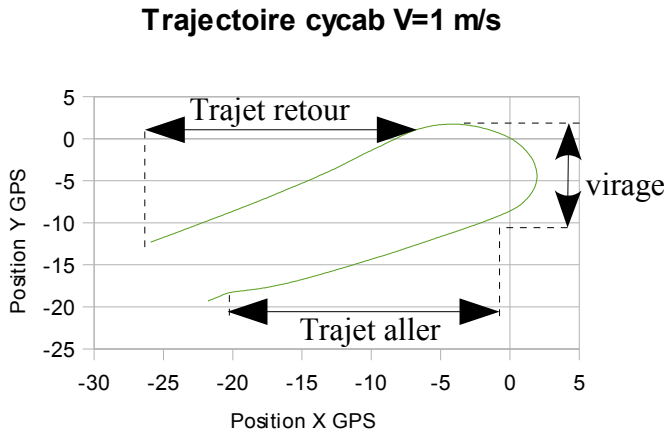
Rôle de l'interpolation linéaire : en prenant 100 points uniformes entre 2 points successifs de la trajectoire de référence (voir chapitre méthode d'interpolation), on diminue fortement l'erreur latérale grâce à une bonne approximation du point le plus proche et par donc cela va permettre de faire converger cette déviation en dessous du centimètre correspondant a la précision du GPS.

De plus ,on peut remarquer que la commande et l'erreur latérale sont fortement bruitées en l'absence de filtrage de la trajectoire de référence et d' interpolation linéaire.
Pour rendre la commande moins bruitée et exploitable pour l'asservissement bas niveau ,on doit aussi filtrer en temps réel la position GPS du cycab.
En effet, le bruits apportes au GPS sont liés aux imprécisions due un manque de satellite, l'atmosphère et les éventuels vibrations du cycab qui désaxe le repère du capteur GPS .

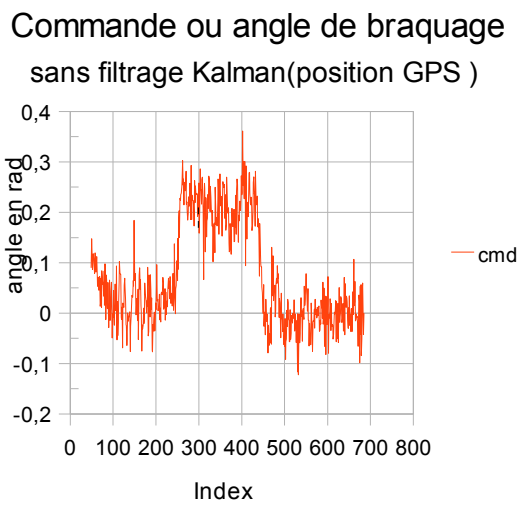
Rôle du filtre Kalman :Il va permettre de reconstruire et estimer les positions et vitesse du capteur.

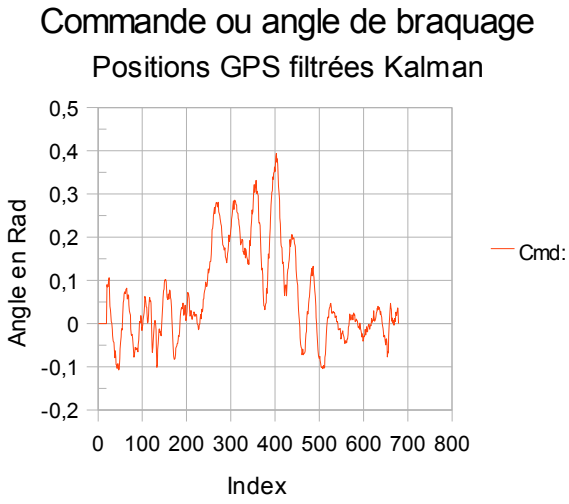
9.1.2)Commande avec interpolation linéaire et avec filtrage Kalman (Trajectoire de référence)

Conditions d'essais: nous utiliserons par la suite cette trajectoire de référence (ci dessous) au cours de nos essais.
Le filtrage Kalman sera appliqué a la fois sur la position GPS en temps réel et la trajectoire de référence . L 'interpolation linéaire de la trajectoire de référence sera aussi intégrée.



9.1.3)Commande sans /avec filtrage Kalman (GPS en temps réel et Trajectoire de référence)





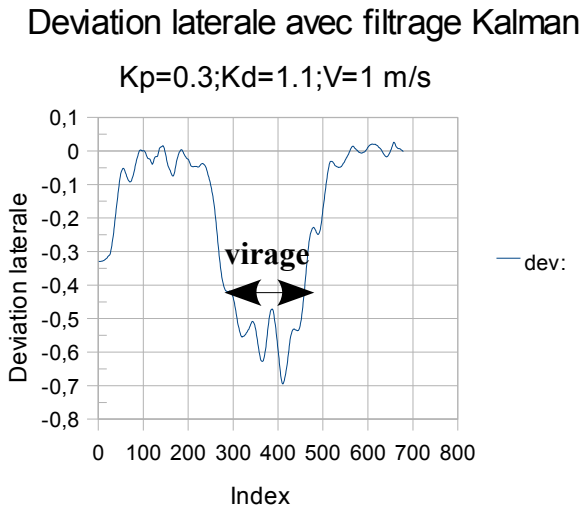
Commentaires:

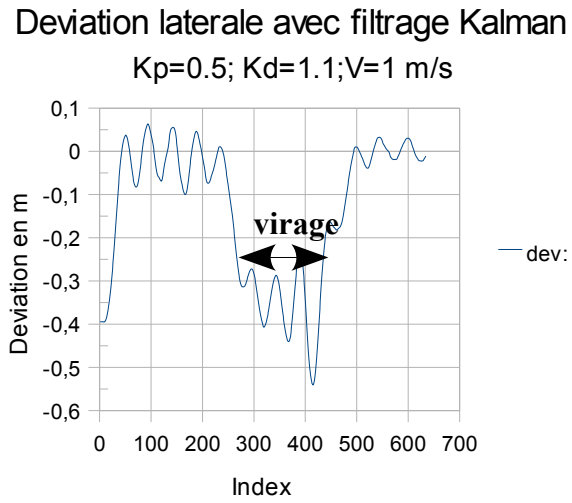
On peut remarquer une nette amélioration sur la forme du signal de commande en présence du filtre Kalman en aval du capteur GPS .
En effet , les a coups sont moins perceptibles par l'asservissement bas niveau des moteurs .

Sachant que la trajectoire de référence a été acquise lors d'un apprentissage et a subie aussi un filtrage ,elle est non régulière et bruitée et engendre des bruits non négligeable dans la commande.
Quand on filtre la position GPS en temps réel , il vaut mieux mettre un R bas que l'on a estimé à 0.001 et un Q élevé car le modèle prédictif à vitesse constante est médiocre.
En fait le modèle prédictif du filtre Kalman est linéaire et ne prend pas en compte le commande u(k) dans l'espace d'état .

9.2) Influence de Kp(gain proportionnelle)

9.2.1)Déviation latérale V= 1m/s; Kd= 1.1 avec *Kp*= 0.5 et *Kp*= 0.3

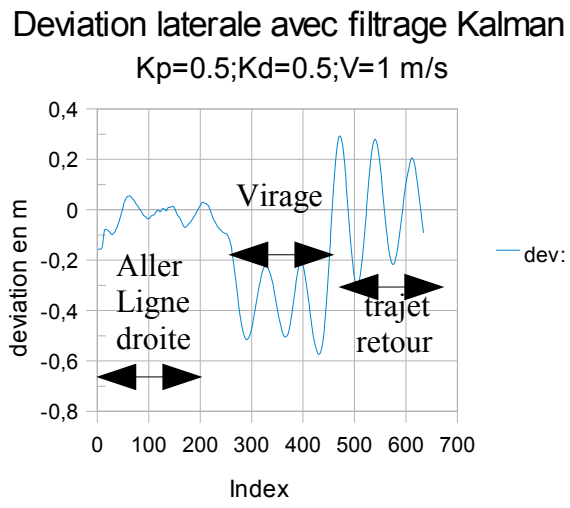


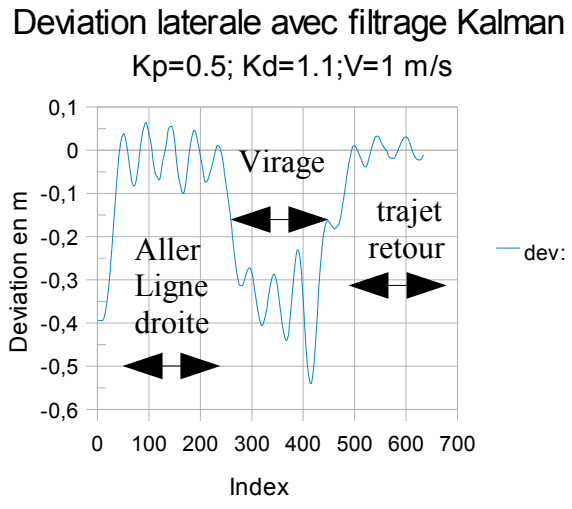


Commentaires: En augmentant Kp c'est a dire en passant de 0.3 à 0.5 , on constate que la déviation latérale est moins important en amplitude mais au détriment d'oscillations .

9.3) Influence de Kd(gain dérivé)

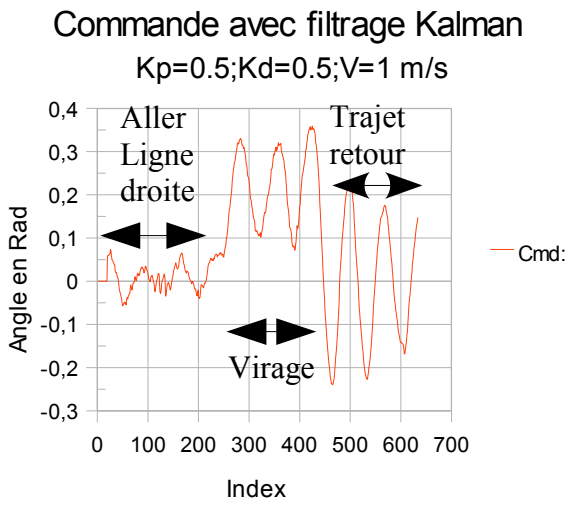
9.3.1)Déviation latérale V= 1m/s; Kp= 0.5vec *Kd*= 0.5 et *Kd*= 1.1

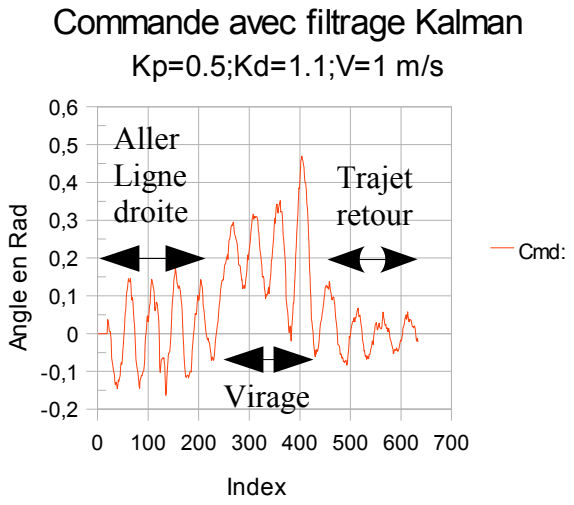




Commentaires: En augmentant , le gain Kd la déviation latérale présente aussi plus d'oscillations . En effet, ceci est due au fait que les valeurs optimales pour le couple Kd et Kp doivent satisfaire la relation de l'équation (**4.7**)cad Kd=2√Kp.

9.3.2)Commande V= 1m/s; Kp= 0.5 avec Kd= 0.5 et Kd= 1.1





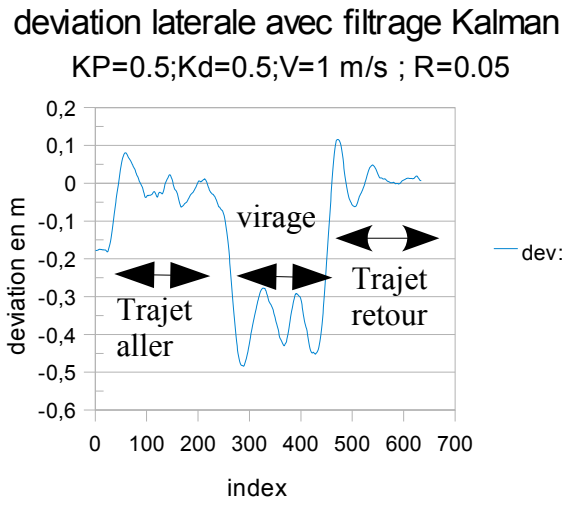
Commentaires:

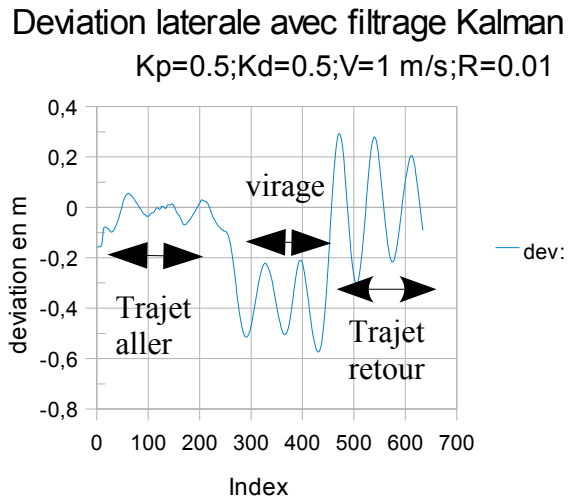
Pour le trajet retour ,on peut remarquer que l'effet dérivé diminue l'amplitude des oscillations avec une meilleur convergence. L'inconvénient est que la commande amplifie aussi du bruit .

9.4) Influence de R (covariance du bruit de mesure)

Conditions d'essais: nous utiliserons une trajectoire de référence V= 1m/s; Kp= 0.5 ;Kd= 0.5 . le filtre sera réglé avec une matrice de covariance mesure R= 0.05.
NB: Le filtrage KALMAN de la mesure GPS en temps réel ne sera pas appliquée.

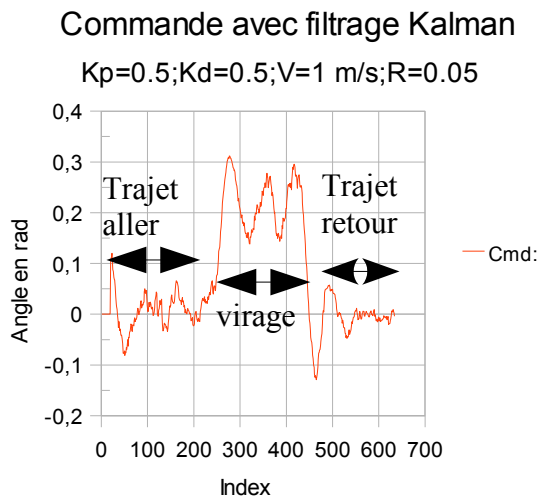
9.4.1)Déviation laterale V= 1m/s; Kp= 0.5 ;Kd= 0.5 ; R=0.05 et R=0.01

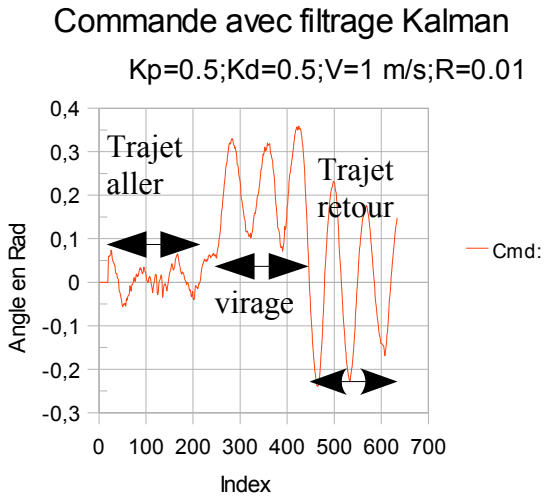




Commentaires:
On constate qu'en augmentant R, on filtre fortement le signal et on atténue les oscillations. De plus la convergence des amplitudes tend plus rapidement vers une valeur nulle(**trajet retour**).
Il est préférable de mettre un grand R pour filtrer la trajectoire de référence mais si on veut filtrer la la mesure de position du GPS en temps réel , il faut choisir un petit ou R optimal afin de ne pas réduire la dynamique du reconstructeur d'état Kalman car cela peut entraîner un retard plus important pour la commande et aussi un dépassement pour la déviation latérale .

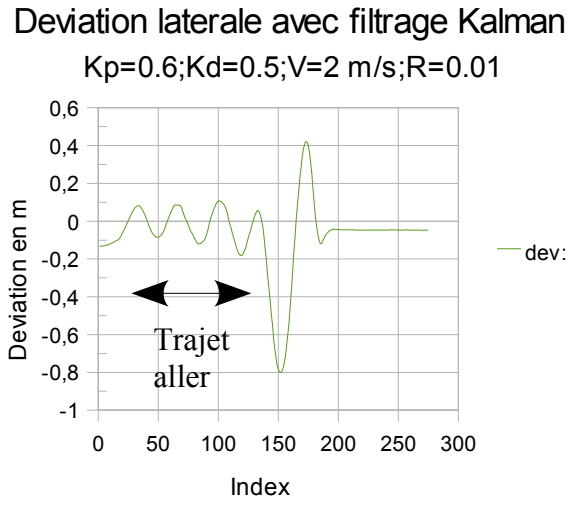
9.4.2)Commande V= 1m/s; Kp= 0.5 ;Kd= 0.5 ; R=0.05 et R=0.01

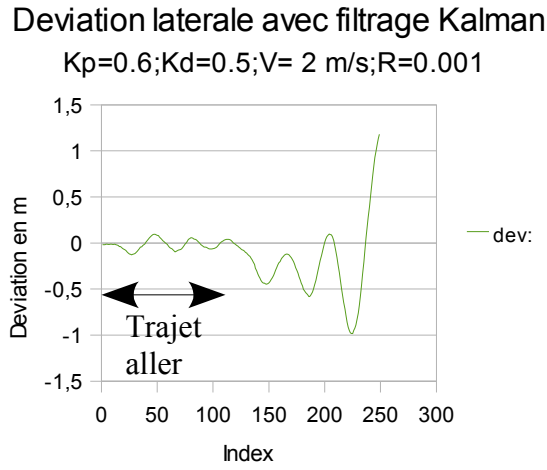




Commentaires:
On constate une différence notoire pour la commande lorsque R =0.05.L'allure est plus bruitée et présente un retard plus important que lorsque R=0.01.
En prenant un R=0.05 ,on filtre fortement les données GPS mais on dégrade la fidélité et la précision du signal GPS donc on commet plus d'erreur au niveau de la commande.

solution :La valeur optimale pour R a été déduit lors des essais en mesurant le bruit de mesure du GPS en trajectoire rectiligne.
Soit ε= 3 cm= 0.03 m soit une variance du bruit de mesure : R= ε^2=0.0009 qu'il faut arrondir à 0.001
Le choix optimal est **Roptimal=0.001 .**





9.4.3)Déviation laterale V= 2m/s; Kp= 0.6 ;Kd= 0.5 ; R=0.01 et R=0.001

Conditions d'essais: nous utiliserons une trajectoire de référence V= 2m/s; Kp= 0.6;Kd= 0.5 . le filtre sera réglé avec une matrice de covariance mesure R= 0.05.
Le filtrage KALMAN de la mesure GPS en temps réel sera également appliquée.
Commentaires: On peut remarquer que la contribution de Roptimal a diminue sensiblement la déviation latérale et on introduit moins de retard au niveau de la commande.

9.5) Influence de la vitesse

Conditions d'essais: nous utiliserons une trajectoire de référence avec filtrage Kalman mais la position GPS du cycab ne sera pas filtrée en temps réel excepté la figure 5 et 6.
Kp= 0.3 ;Kd= 1.1 et V= 0.5 ; V= 1 m/s ;V= 2 m/s ;
le filtre sera réglé par avec une covariance de mesure R= 0.01.

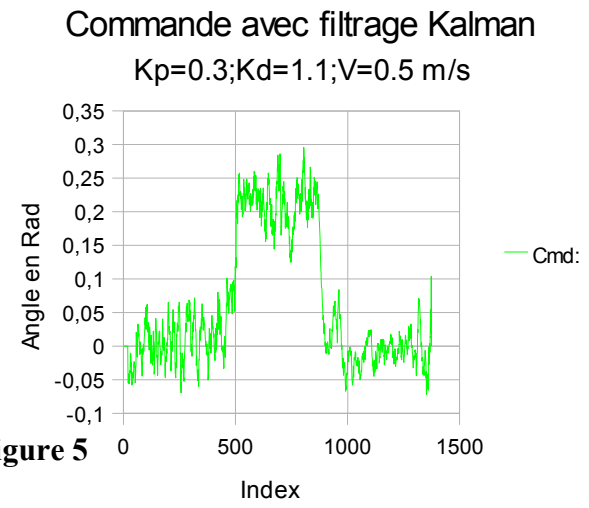


figure 5

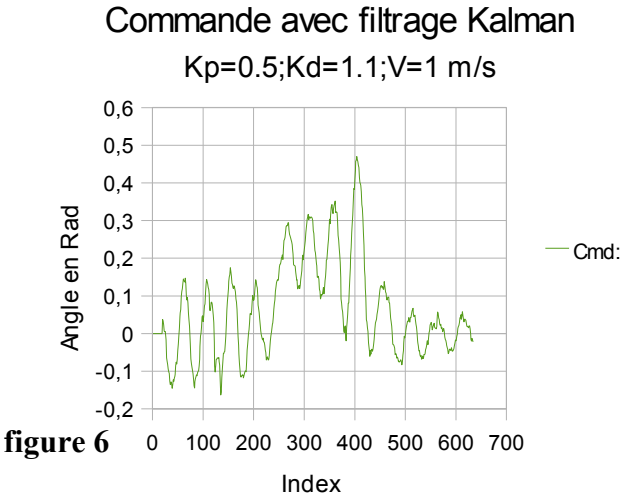
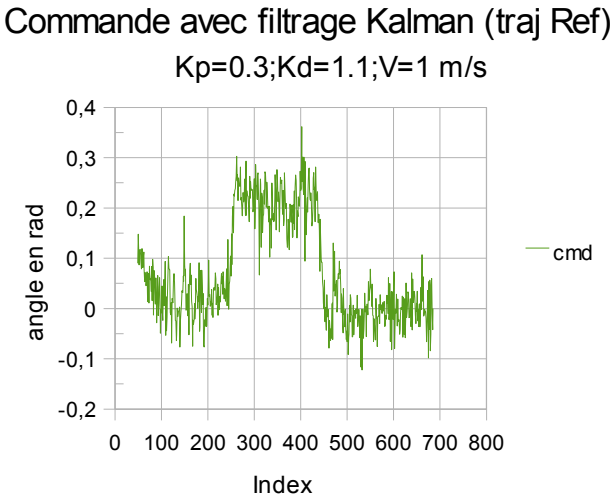
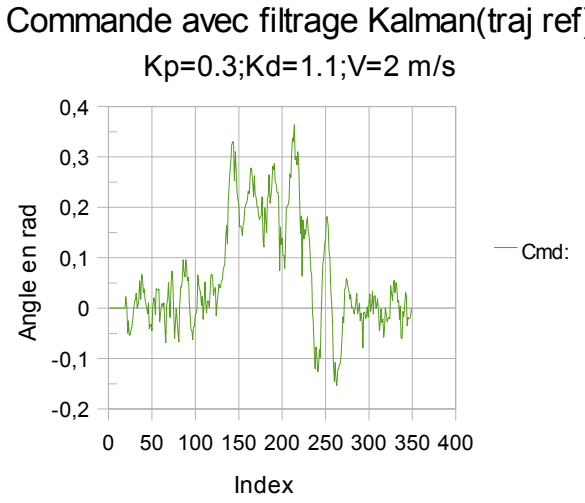


figure 6

commentaires : On constate que lorsque qu'on double la vitesse et en mettant un Kp=0.5 la commande a due mal à converger. Pour avoir une commande avec moins d'oscillations ,il faut réduire Kp ou mettre moins de proportionnel





Explications: Sachant que le GPS est échantillonné tous les 100 ms , plus le cycab va vite et meilleur sera l'approximation du point le plus proche de la courbe et moins il est nécessaire d'interpoler et de filtrer en temps réel la position du cycab.
A contrario, si la vitesse du cycab est faible V=0.5 m/s (ci dessus) , plus le signal récupéré par le GPS est bruité et plus l'erreur latérale due au point le plus proche est élevée et bruitée .

10.IMPLEMENTATION & AMELIORATIONS

Implémentation :

10.1)problèmes rencontres :

Lors de l' implémentation , les points bloquant et qui ont été très couteuse en temps ont été lié essentiellement aux **problèmes trigonométriques** .

- **Fonction ATAN:**

IL ne fallait pas se mélanger entre la fonction ATAN() qui calcule l'arc-tangente entre $-\pi/2$ et $+\pi/2$ alors que ATAN2 calcule l'arc-tangente entre $-\pi$ et $+\pi$.
En effet la détermination du cap de la trajectoire de référence et du cycab doivent utiliser la fonction ATAN2. Soit :

- **Fonction Modulo:**

A chaque soustraction ou d'addition d'angle , il fallait bien faire attention que l'angle résultant soit modulo 2π.en utilisant la fonction remainderf().

- **la détermination de la courbure :**

Le problème ne ne pose pas pour une trajectoire synthétique ou calculée car les échantillons des positions sont déterministes alors qu'une trajectoire réelle comportait beaucoup de bruits pouvant fausser la courbure .
La 2 ème approche en utilisant les fonctions polynomiales **spline** m'a fait perdre beaucoup de temps. mais a été très bénéfique pour le projet .

10.2Améliorations:

filtre KALMAN

- Le modèle du filtre adopté est linéaire c'est à dire qu'on suppose que la mesure ,l'entrée (nulle pour notre projet) et les variables d'états sont tous linéaires.
  De plus , on admet aussi que le bruit de mesure et le bruit du modèle sont indépendants .
  On aurait pu intégrer la commande u homogène a une accélération dans le filtre Kalman pour une meilleur prédiction et augmenter sa robustesse tant en dynamique ,qu'en précision .
- Pour améliorer ce filtre, on pouvait aussi le remplacer par la version étendu ou utiliser l'EKF (extended kalman filter) qui suppose que la mesure et le vecteur d'état sont des combinaisons non linéaires. Cette approche permet de linéariser la mesure en utilisant les développement limitées du Théorème de Taylor .
- Ou utiliser le «filtre infini» ou H¥ qui est une alternative au filtre KALMAN .Ce filtre suppose qu'il y a une corrélation entre le bruit de mesure et celui du modèle.

Temps de calcul :

Le correcteur de trajectoire élabore et envoie une commande tous les 100 ms c'est à dire qu'entre 2 positions du GPS le calculateur passe la majorité de son temps à attendre le réveil de la prochaine mesure GPS.Pour permettre une meilleur robustesse et avoir une erreur latérale et angulaire quasi nulle, on peut soit utiliser l'odométrie comme information de retour de mesure ou faire une prédiction de la position au moyen d'estimateur et ce à une fréquence de 100Hz .Cette fréquence de calcul est limitée par les contraintes temps réel du système et dépend essentiellement de la latence de l'asservissement PID des moteurs.

BILAN DU STAGE

Le stage m'a permis de tester une loi de commande et qui a pu faire ses preuves sur le terrain. Faute de temps , la loi de commande avec prise en compte de la courbure de la trajectoire a été uniquement simulé avec des résultats très convaincants (voir ANNEXES).

Une étape à été amorcée pour ce projet et des améliorations peuvent être apportées sur les points énoncées précédemment tel que sur le filtre Kalman, l'interpolation de la trajectoire pour la détermination de la courbure et aussi sur la puissance de calcul de la commande .

Sur le plan personnel , ce stage m'a beaucoup intéressé et motivé car j'ai pu me rafraichir et découvrir des domaines très diverses tel que l'automatique avancée , la programmation en C++, la théorie des splines pour l'interpolation .
Le logiciel SCILAB a aussi été un outil indispensable et très présent durant ce stage .

Les ressources mis à disposition tel que la bibliothèque de L'INRI A et l'accès aux articles IEEE m'ont permis d'enrichir mes connaissances.

De plus , j'ai pu apprendre une certaine méthodologie de travail pour mener à bien ce projet.

ANNEXES

Hardware du Cycab

Infos PC

- Carte Mère :

Cycab Gris	*Cycab Rose*
ROCKY-4783EV	ROCKY-568SEV

- Carte RS422 pour liaison série à 500 kbps (interface Sick) : QUATECH-DSC-200
- Carte CAN de type PCI : Adlink PCI7841

Infos Robot

- ID Noeud :

Cycab Gris	*Cycab Rose*
f555 = 0x4001	
r555 = 0x4000	

- Codeur Direction/Roues :
  - Fabricant : IVO Industries
  - Types:

Direction	*Roues*
Codeur absolu parallele GA241	Codeur incremental GI338
Resolution : **`13 bits`**	Resolution : **`500 imps`** par tour

- Variateurs de puissance PWM :
  - Fabricant : Advanced Motion Controls
  - Type : 50A8DDx pour les moteurs de traction et 30A8DDx pour le vérin de direction
  - documentation
  - Freq hachage = 20 kHz
- Variateurs de puissance analogiques :
  - Fabricant : Curtis
  - Type : 1237-41xx pour les moteurs de traction
  - documentation
  - tension de commande : +/- 10Volts

Parametres Moteur

Vitesse max moteur	**`2800 rpm`**, soit **`3,62 m.s-1`** (ou **`13 km/h`**)
Resolution roue moteur	**`2000`** tops (4 * 500 imp) par tour moteur après décodage quadrature
Rapport de réduction	**`17`**
Pour un déplacement de 1 m	(2000 * 17) / (0.42 * PI) = **`25768 tops`**

Paramètres physiques

Diamètre des roues	`420mm`
Distance entre 2 roues d'un meme train	`1,18m`
Distance entre train avant et train arriere	`1,21m`
Angle de braquage max	`28,75degres`

- MPC555 Documentation
  - http://cycabhdk.gforge.inria.fr/documents.html
- Alimentation : 8 batteries Tudor TD 60 (12V - 60Ah)

Infos Capteurs exteroceptifs et autres interfaces

- Joystick 2 axes + switch :
  - fabricant : CHProducts
  - modèle : HFX-22S12-034
  - documentation : HFX-22S12-034
- Télémètre à balayage laser :
  - fabricant : SICK
  - modèle : LMS200
  - ouverture angulaire : 180 degrés
  - résolution angulaire : 0,25 degré
  - portée : 80 mètres
  - documentation technique : LMS200
  - sur le pc embarque : accessible sous **`/dev/ttyS4`**
- GPS centimétrique RTK :
  - fabricant : Magellan Navigation (ex Thales)
  - modèle : Z-Max
  - dual frequency (L1 et L2)
  - mode RTK (Real Time Kinematics)
  - précision 10mm + 1ppm RMS
  - rafraîchissement des mesures : 10 Hz
  - documentation : Z-Max, ZFamily
- Camera monostereo EVT Marlin
- Camera stereo SVS Videre Design

Configuration de l'OS du PC embarqué du Cycab

Cycab blanc/gris : **bcycab**

Cycab rose : **rcycab**

Caractéristiques générales

Nom machine	bcycab	rcycab
Adresse IP	194.199.21.107	194.199.21.106
CPU	Pentium IV 1.8 Ghz	Pentium IV 2,4 GHz
Mem - Disk	230 Mb	256 Mb - 40 GB
Ecran Tactile	ICP Electronics DM-65G acheté chez http://www.ipo-sa.com Resolution : 640x480	ICP Electronics DM Series 64T/T-R064B avec Capteur AMT Résolution : 640x480 Driver : Penmount
OS supporté	Noyau Linux 2.4.22 RedHat 9.0	Noyau Linux 2.4.22 RedHat 9.0
	Noyau Linux 2.6.22 Ubuntu 7.10	Noyau Linux 2.6.22 Ubuntu 7.10
RT patch	rtai-24.1.13	rtai-24.1.13
	aucun patch	aucun patch
Adresse de noeuds 555	f555 = **0x4001** r555 = **0x4000**	
Vitesse bus CAN	**500 Kbps**	
Syndex	v6.7.0	
Config	config-2.4.22-adeos configRTAI	

Guide de Manipulation sur le Cycab

BUT:

- PC embarque sur le Cycab : log des donnees odometrie, du sick, du GPS
- PC portable : log des donnees camera ou autre capteur installe sur ce poste

Etapes :

Synchroniser les 2 PCs :

- Relier les 2 PCs avec un cable croise
- Modifier la configuration du serveur `ntp` pour que les 2 postes soient le + synchronises possibles
    - Sur le PC portable

- Configurer le fichier **/etc/ntp.conf**

```
server 127.127.1.1
fudge 127.127.1.1 stratum 8 refid NIST
restrict IPCycabPC
```

    - Redemarrer le demon ntp **/etc/init.d/ntpd restart**
    - Attendre un reach > 7 via **watch ntpq -c lpeers**
- Sur le PC embarque du Cycab (rcycab : IP = **194.199.21.106**)
    - Arreter le demon ntp **/etc/init.d/ntpd stop**
    - Lancer **ntpdate -v -b IPPCPortable**
    - Redemarrer le demon ntp **/etc/init.d/ntpd restart**

Verifier que le PC portable gère convenablement la camera

- Si il s'agit d'une camera firewire, utiliser **coriander**
- En fonction de la reponse de **coriander**, vous serez peut-etre amene a configurer correctement
    - **chmod a+rwx /dev/video1394**
    - **chmod a+rwx /dev/raw1394**

Logguer les donnees en utilisant le middleware **Hugr**

- Sur le PC portable (bleu)
    - Lancer hugr en mode reseau (pour recuperer les donnees du pc embarque)
        - **hugrstore -n -g manipcycab --name aliasPCPortable -s 8000000**
    - Lancer l'acquisition video via hugr
        - **hugrvideo**
    - Lancer la visualisation des images acquises via hugr
        - **hugrvideoplay**
    - Logguer les donnees (video provenant du PC portable, et GPS, odometrie et Sick provenant du PC embarque) acquises via hugr
        - **hugrlog -o /tmp/logcamera camera Donnees_GPS cycab_state LMS291Conf sickLMS291**
    - Une fois les donnees acquises, on peut les rejouer via hugr
        - **hugrreplay /tmp/logcamera**
    - Pour obtenir des infos sur un fichier de log acquis via hugr
        - **hugrreplay --info nomfichierlog**
- Sur le PC embarque sur le Cycab (rcycab)
    - Lancer hugr en mode reseau
        - **hugrstore -n -g manipcycab --name aliasPCembarque**
    - Lancer les applications pour communiquer avec le GPS, le Sick (sur le port serie correspondant) et recuperer l'odometrie du robot via hugr
        - **GPSServer**
        - **SickServer /dev/ttyS2**
        - **CycabServer**

GPS

Caractéristiques :

Parameter	Specification
GPS	Static, Rapid Static - 24 parallel channels all-in-view - L1 C/A code and carrier - L1/L2 P-code, full wavelength carrier - Z-Tracking - Multipath mitigation - Integrated real-time WAAS/EGNOS - Update rate: 10 Hz - Protocol: NMEA 0183
Accuracy (1-2)	Static, Rapid Static - Horizontal 0.005 m + 0.5 ppm (0.016 ft + 0.5 ppm) - Vertical 0.010 m + 0.5 ppm (0.033 ft + 0.5 ppm) Post-processed Kinematic - Horizontal 0.010m + 1.0 ppm (0.033 ft + 1.0 ppm) - Vertical 0.020 m + 1.0 ppm (0.065 ft + 1.0 ppm)
Real-Time Performance (1-2)	SBAS (WAAS/EGNOS) (rms) - Horizontal: < 3 m (10 ft) Real-Time DGPS position < 0.8 m (2.62 ft) Real-Time Kinematic Position (fine mode) - Horizontal 0.010m + 1.0 ppm (0.033 ft + 1.0 ppm) - Vertical 0.020 m + 1.0 ppm (0.065 ft + 1.0 ppm) Instant-RTK Initialization - 99.9% reliability - Typical 2-second initialization for baselines < 20 km

Équations du filtre Kalman

On considère un processus linéaire discret décrit par l'équation :

$$\begin{cases} x(k+1)=A.x(k) + B.u(k) + G.w(k) \\ y(k)=C.x(k) + v(k) \end{cases}$$

w(k) et v(k) sont respectivement des bruits blancs non corrélées lies au **processus** et à la **mesure**

$$\begin{cases} E[w(k)]= 0 \quad ; \quad E[v(k)]= 0 \\ E[w(k).w(i)T]=\delta(k-i).Q(k) \quad E[v(k).v(i)T]=\delta(k-i).R(k) \end{cases}$$

Q(k) représente la covariance du bruit du modèle R(k) représente la covariance du bruit de mesure

Notations :

l'etat :

X(k)

l'état prédit :

$\hat{X}(k \mid k-1)$

L'état filtré :

$\hat{X}(k \mid k)$

Equations du filtre Kalman :

$$\hat{X}_{k|k-1} = A\hat{X}_{k-1|k-1} + Bu_k \quad (1)$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q \quad (2)$$

$$J_k = P_{k|k-1}C^T(CP_{k|k-1}C^T + R)^{-1} \quad (3)$$

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + J_k(Y_k - C\hat{X}_{k|k-1}) \quad (4)$$

$$P_{k|k} = (I - J_kC)P_{k|k-1} \quad (5)$$

Etape 1 :La mise a jour du temps (the time update"Predict")

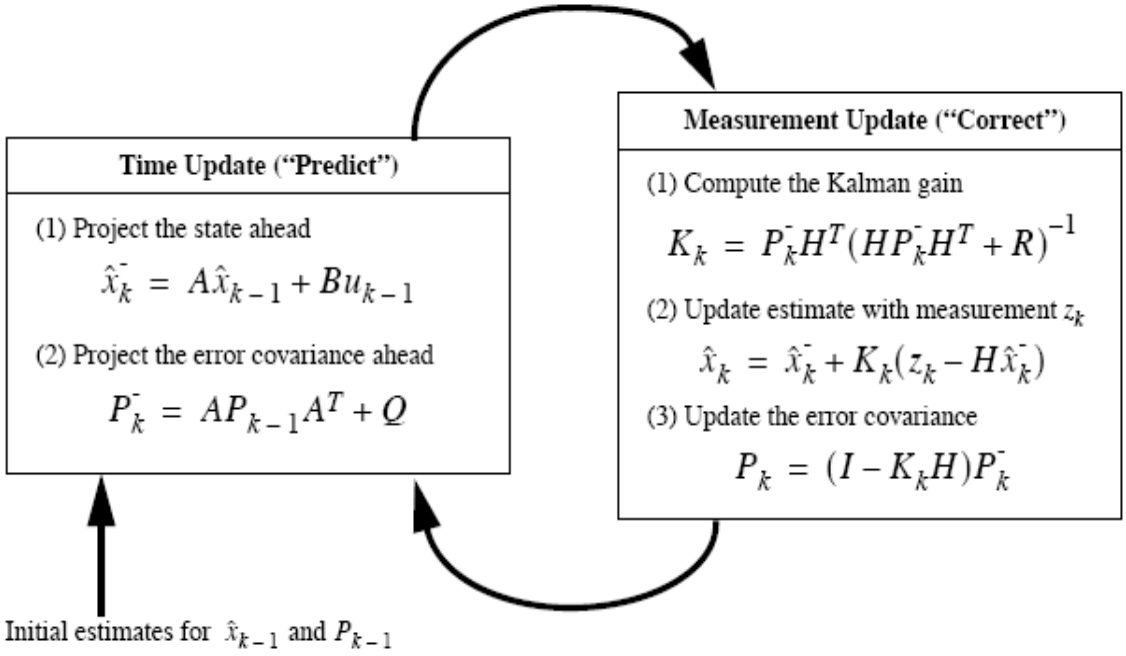
La 1ere équation calcule à «priori» l'estimation du vecteur d'état. Elle dépend de l'état précédent et de l'entrée du système
La 2eme équation met a jour «à priori»la matrice P de l'estimation l'erreur de covariance .

Etape 2:La mise a jour des mesures (Measurement update)

ce sont les équations (3), (4), (5) qui permettent de réaliser cette mise a jour des mesures
L'équation 3 calcule le gain Kalman J permettant de minimiser «à posteriori»la matrice P de l'erreur de covariance

L'équation 4 calcule "à posteriori" l'estimation du vecteur d'état pour la nouvelle mesure .
Pour cela ,elle compare la vraie mesure avec la mesure prédite en utilisant la différence appelé "***innovation***' ou résidu ' ε = (Yk – C.Xˆk-1) .

L'équation 5 calcule a "posteriori" la matrice P de l'erreur de covariance
Description des étapes



<u>Réglages des paramètres de l'algorithme du filtre Kalman :</u>

<u>Matrice R:</u>
R représente *la matrice de covariance du bruit de mesure R= E(vk. (vk)T)* .
Elle détermine la quantité d'informations liées aux échantillons doit être utilises.
Si R est grand , cela signifie que les mesures ne sont pas très précises.
Si R est petit , le filtre recueillera plus d'informations et suivra exactement les mesures.

<u>Matrice Q:</u>
Q représente *la matrice de covariance du bruit modèle/ entrée* Q= E(wk. (wk)T)
avec E: espérance mathématique et T: transpose
Il y a un compromis entre la poursuite et le bruit en sortie , en effet si Q est grand , le sortie sera plus bruyante car l'estimation va "fluctuer " plus longtemps pendant une période d'échantillonnage.
Et si Q est petit , plus la poursuite sera fidèle.

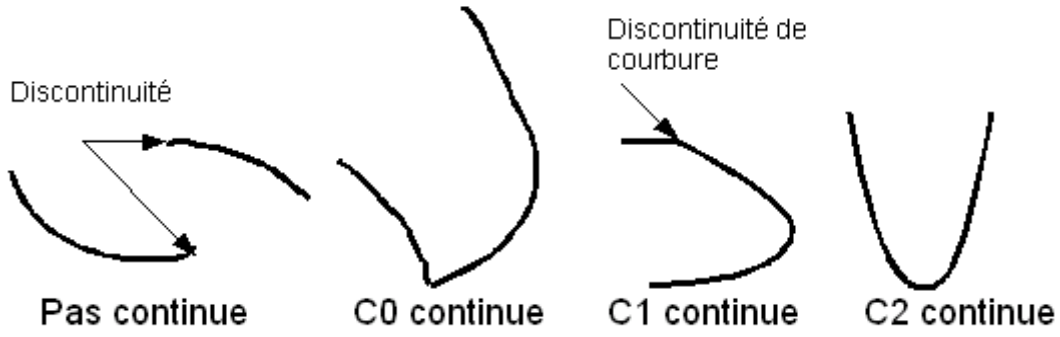
Interpolation cubique spline

Elle a la particularité de garantir une forme régulière et lisse de la courbe et d'être continue C² (voir Annexes)
pour un jeu de donnes {xi} de n+1 points , on peut construire une spline cubiques avec n fonctions polynomiales cubiques par morceaux entre les points

définition de la continuité :

soit f un un intervalle réel ,

$$f : I \to \Re \text{ et } a \in I$$
$$\text{la fonction } f \text{ est dite en } a \text{ si :}$$
$$\forall \epsilon > 0 \ \exists \eta > 0 \ \forall x \in I \ [|x-a| \leq \eta \Rightarrow |f(x)-f(a)| \leqslant \epsilon]$$



C⁰ continue : les 2 courbes sont continue au 0 ordre car ils ont la même de position .
C¹ continue : les 2 courbes sont continue au 1er ordre car ils ont la même tangente .
C² continue : les 2 courbes sont continue au 2 ème ordre car ils ont la même courbure(**courbe lisse**)

S(x) représente la fonction spline interpolant la fonction f .

$$S(x) = \begin{cases} S0(x),\ x \in [x0,x1] \\ S1(x),\ x \in [x1,x2] \\ \quad \ldots\ldots \\ Sn\text{-}1(x),\ x \in [xn\text{-}1,xn] \end{cases}$$

Les propriétés sont :

- propriétés d'interpolation , $S(xi) = f(xi)$
- la jointure des splines S(i-1)(xi)=Si(xi), i=1,....,n-1
- la continuité en C² S'i-1(xi)=Si'(xi) et S"i-1(xi)=Si"(xi), avec i=1,..,n-1

avec S'(xi) étant la dérivé 1ere et S"(xi) la dérivé seconde

comparaison des différents types de splines cubiques :

Type	**Points de contrôles**	**Continuité**	**Interpolation**
Hermite	oui	C¹	oui
Bezier	oui	C¹	oui
Catmull-Rom	oui	C¹	oui
Natural	non	C²	oui
B-splines	oui	C²	non

DETERMINATION DE LA COURBURE

1)algorithme du cercle circonscrit:

```
void circumCircleCenter(float x1, float y1, float x2, float y2, float x3, float y3, float &x, float &y)
  {
   float v,m1, m2, m3, n1, n2, n3;

   m1 = 2.*(x3-x2); m2 = -2.*(y2-y3); m3 = (y2-y3)*(y2+y3)-(x3-x2)*(x2+x3);
   n1 = 2.*(x3-x1); n2 = -2.*(y1-y3); n3 = (y1-y3)*(y1+y3)-(x3-x1)*(x1+x3);
   v=1./(n2*m1-n1*m2);
  x = v*(n3*m2-n2*m3);
  y = v*(n1*m3-n3*m1);
 }
```

1)algorithme par les fonctions cubiques splines et B-splines:

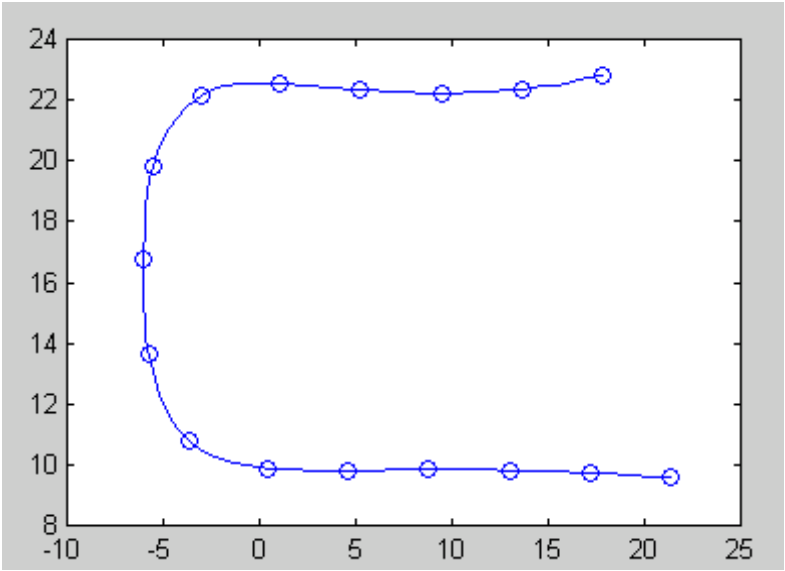
en utilisant les fonctions cubiques par morceaux et en le convertissant en B-splines

```
b=[];
fin=500 ;

fid=fopen('Trajectoire_ref_filter31.log','r');%read trajectory file

b=fscanf(fid,'%g %g %g %g %g ',[5,fin])
b=b'

X=b(:,1)
Y=b(:,2)

index=1;
Np=300
saut=20 % spacement of pts
%Np=6
y_d=X(index:saut:index+Np) %invers abcissa and ordinate
x_d=Y(index:saut:index+Np)

x=x_d' ;
y=y_d' ;

Np=Np/saut ;

xy = [x;y];

t = linspace(0,Np,Np+1)
cv = csapi(t,xy);
 fnplt(cv), hold on, plot(x,y,'o'), hold off  %display trajectory
spb = fn2fm(cv,'B-')  % convert to B-form spline
int=fnbrk(cv,'interval')%retrieves intervall of t
t = linspace(0,Np,5*Np)
% curvature process of B=spline
dsp = fnder(spb); dspt = fnval(dsp,t); ddspt = fnval(fnder(dsp),t);
 %(t) = ||gamma'(t)×gamma''(t)|| /|| gamma'(t)||^ 3
kappa = (dspt(1,:).*ddspt(2,:)-dspt(2,:).*ddspt(1,:))./...
(sum(dspt.^2)).^(3/2);
[min(kappa),max(kappa)]

figure %new figure
plot(t,kappa)%display curvature
title('curvature B- spline') %
```

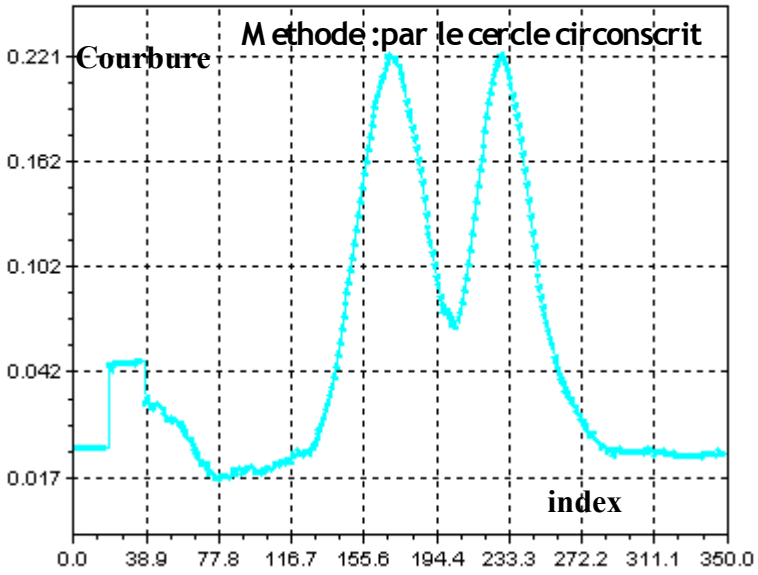
SIMULATION DE LA COURBURE

trajectoire de référence

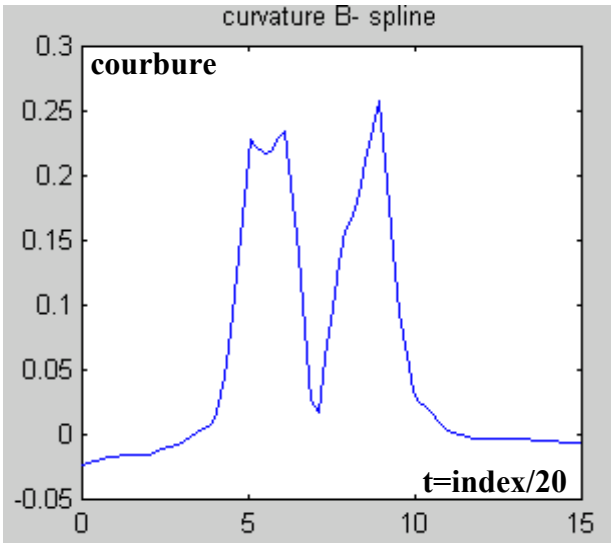
les points sont espacés tous les 20 échantillons GPS



simulation sous SCILAB



simulation sous MATLAB



Parametric Curves

Curves and surfaces can have explicit, implicit, and parametric representations. Parametric representations are the most common in computer graphics.

Reparameterization

Parameterizations are in general not unique. Consider the following parameterizations for a line:

```
L(P0,P1) = P0 + u(P1-P0),   u = [0...1]
L(P0,P1) = v(P1-P0)/2 + (P1+P0)/2,   v = [-1...1]
```

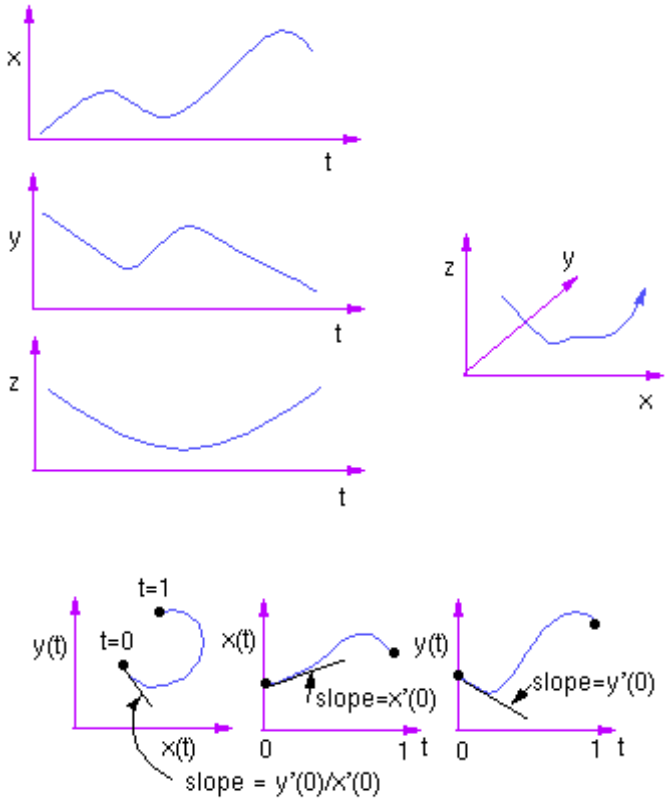
Parameterizations can be changed to lie between desired bounds. To reparameterize from u = [a...b] to w = [0...1], we can use w = (u-a)/(b-a), which gives u = w(b-a) + a. Thus, we have:

```
P(u), u = [a...b]    =    P(w(b-a)+a), w = [0...1]
```

Parametric Cubic Curvess

Cubic curves are commonly used in graphics because curves of lower order commonly have too little flexibility, while curves of higher order are usually considered unnecessarily complex and make it easy to introduce undesired wiggles.

A parametric cubic curve in 3D is defined by:



$$x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$y(t) = b_3 t^3 + b_2 t^2 + b_1 t + b_0$$
$$z(t) = c_3 t^3 + c_2 t^2 + c_1 t + c_0$$

Usually, we consider t = [0...1].

A compact version of the parametric equations can be written as follows:

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

$$x(t) = T \cdot A$$

Similarly, we can write

y(t) = T B

z(t) = T C

Each dimension is treated independently, so we can deal with curves in any number of dimensions.

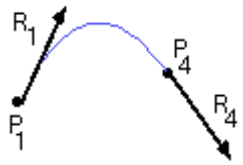
The derivatives of the curve with respect to t can be expressed as follows:

x'(t) = [3t^2 2t 1 0] A

It is often convenient to think of the parameter t as being time in order to visualize some of the properties of the curve. The derivatives also have an intuitive interpretation in the cartesian space of the curve:

Hermite Curves

As a second example, let's look at Hermite curves. Hermite curves are defined by two points and two tangent vectors.



Let's derive the equation for Hermite curves using the following geometry vector:

```
G_h = [ P1 P4 R1 R4 ]^T
```

As before, we'll express the curve as:

```
x(t) = T A_h
     = T M_h G_h
```

The constraints we'll use to define the curve are:

```
x(0)  = P1 = [ 0 0 0 1 ] A_h
x(1)  = P4 = [ 1 1 1 1 ] A_h
x'(0) = R1 = [ 0 0 1 0 ] A_h
x'(1) = R4 = [ 3 2 1 0 ] A_h
```

Writing these constraints in matrix form gives:

```
G_h = B_h A_h
A_h = inv(B_h) G_h
x(t) = T A_h
     = T inv(B_h) G_h
     = T M_h  G_h
```

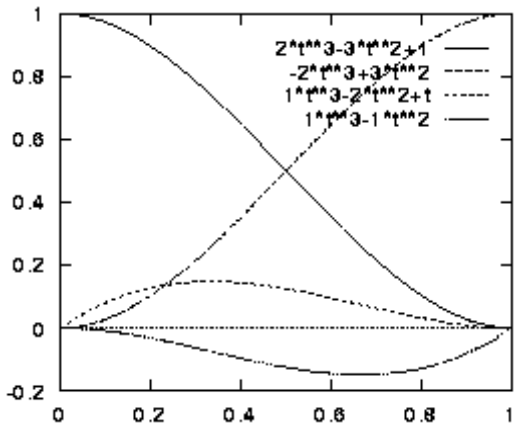
The inverse of B_h is thus defined as the basis matrix for the hermite curve.

```
       [  2 -2  1  1 ]
M_h =  [ -3  3 -2 -1 ]
       [  0  0  1  0 ]
       [  1  0  0  0 ]
```

As before, the basis functions are the weighting factors for the terms in the geometry vector, and are given by the product T M_h. Thus, for basis functions for Hermite curves are

```
f1(t) =  2t^3 - 3t^2 + 1
f2(t) = -2t^3 + 3t^2
f3(t) =   t^3 - 2t^2 + t
f4(t) =   t^3 -  t^2
```

These basis functions look as follows:



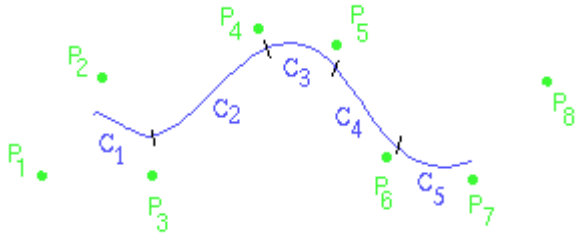
Summary of Hermite Curves

- offer local control
- offer C1 continuity
- interpolates (some) control points

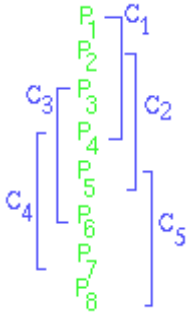
Splines

- *splines* are cubic curves which maintain C2 continuity.
- *natural spline*
    - interpolates all of its control points.
    - equivalent to a thin strip of metal forced to pass through control points
    - no local control
- *B-spline*
    - local control
    - does not interpolate control points

The following is an example of a five-segment B-spline curve (although this is simply a hand-drawn example). The points which indicate the ends of the individual curve segments and thus the join points are known as the *knots*.



Each curve segment is determined by four control points, as follows:

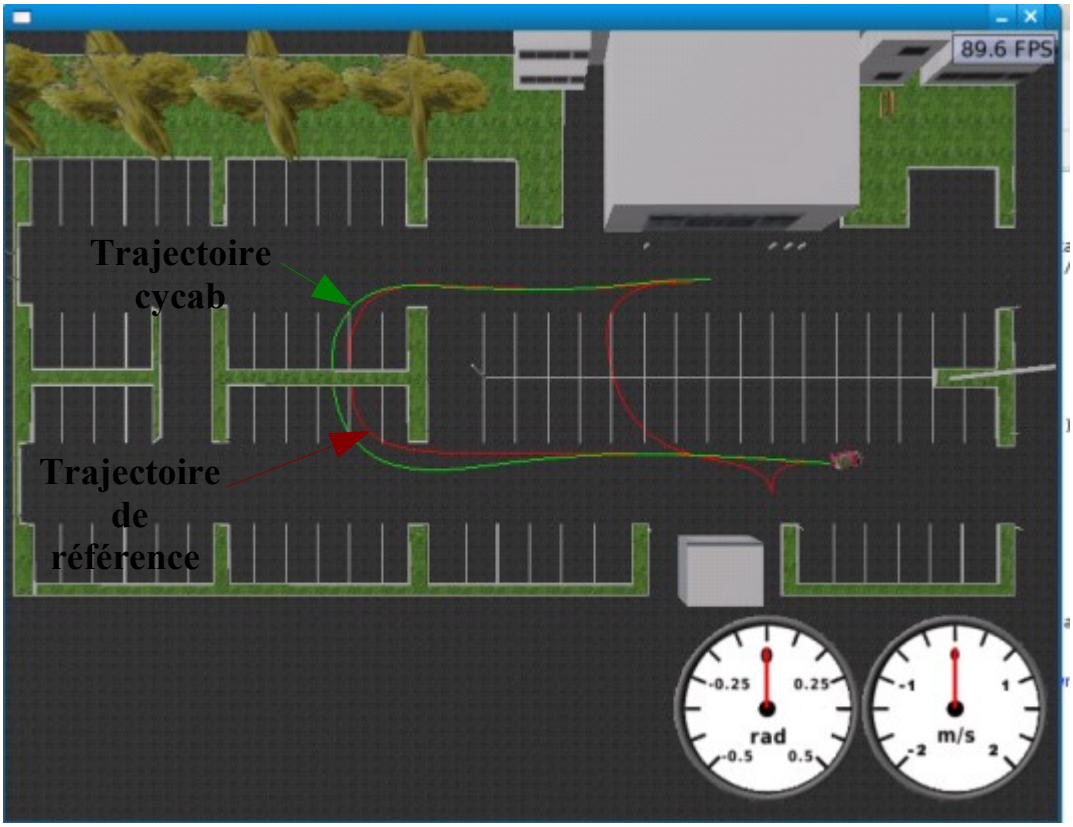


B-spline curves are defined by a basis matrix, just like the other types of cubic curves. We shall not discuss the derivation of this matrix here.

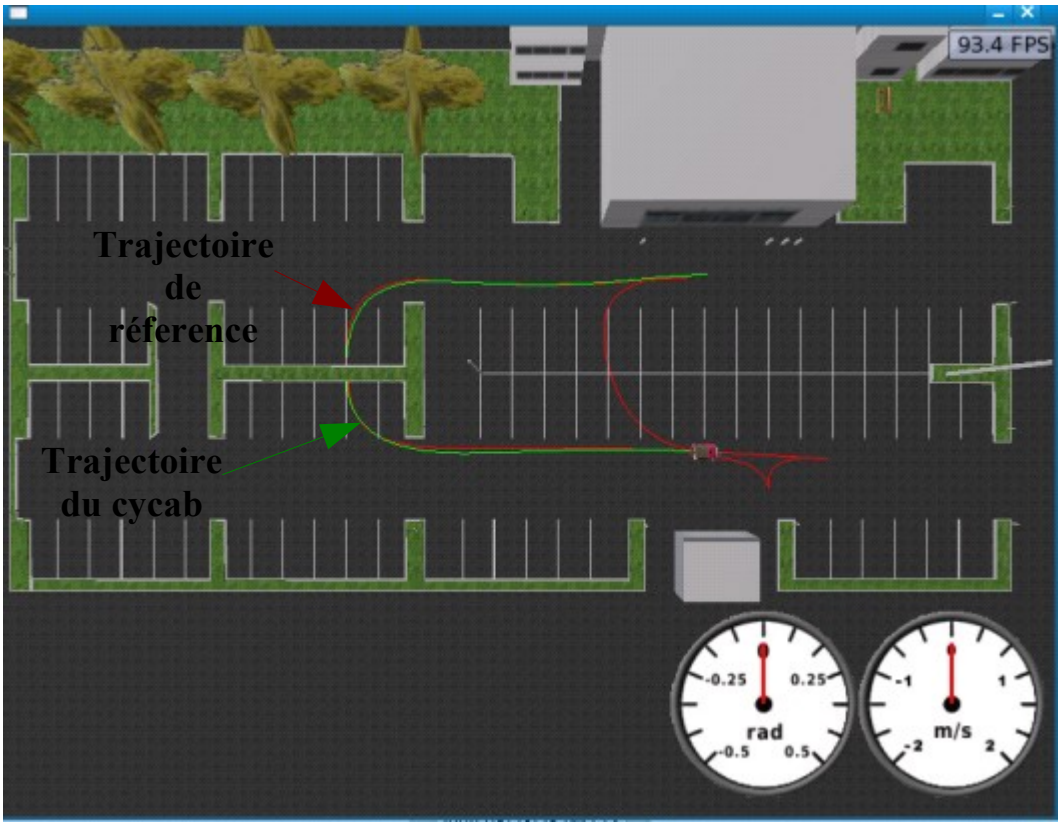
$$M_{bspline} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

SNAPSHOT sous MGENGINE

Loi de commande sans courbure ou c(s)=0 et sans interpolation



Loi de commande avec courbure c(s)≠0 et interpolation



BIBLIOGRAPHIE

[1] Rapport technique « Le Cycab de l'INRIA RhôneAlpes» , Gérard Baille, Philippe Garnier, Hervé Mathieu, Roger PissardGibollet,Avril 1999.

[2] Feedback Control of a Nonholonomic Car-like Robot A. De Luca G. Oriolo C. Samson

[3]Thomas Hellström.Autonomous Navigation for Forest Machines – a Pre-Study

[4] Wit, J. S., "Vector Pursuit Path Tracking for Autonomous Ground Vehicles," Ph.D. thesis, University of Florida, 2000.

[5] C. Samson. Control of chained systems. Application to path following and timevarying point stabilization of mobile robots. IEEE Transactions on Automatic Control,40(1):64–77, 1995.

[6] Automatic guidance of a farm tractor along curved paths, using a unique CP-DGPS, B.THUILOT, C.CARIOU,L.CORDESSES,P.MARTINET .

The Zodiac. *Theory of robot control.* C. Canudas de Wit, B. Siciliano, G. Bastin editors, Springer-Verlag, Berlin, 1996.

M. O'Connor, G. Elkaim, T. Bell, and B. Parkinson.Automatic steering of a farm vehicle using GPS.In Proceedings of the International Conference on Precision Agriculture 1996, volume 3, pages 767–777, 1996.

P. Morin. Feedback control of nonholonomic mobile robots. I*n 5th Summer School on Image and Robotics,* 2004.

C. de Boor. A practical guide to splines. Springer-Verlag, 1978.

Spline Toolbox™ User's Guide by C. de Boor and The MathWorks, Inc.

Helene Mörtberg .control and dynamic Modeling of an autonoumous ground vehicle january 2006