

RAPPORT DE STAGE DE FIN D'ÉTUDES

FSIMACS - 3ème année

Simulation de trajectoires de véhicules

Laboratoire IMARA

Auteur: AIT MEBAREK Hayet

Tuteur: M. DE LA FORTELLE Arnaud

16 septembre 2005

Table des matières

Remerciements	4
Introduction	5
I Présentation de l'entreprise et du stage	7
1 Présentation de l'INRIA	8
1.1 Domaines de recherche	8
1.2 Le projet IMARA	9
1.2.1 Domaine d'application	9
1.2.2 Objectifs	10
1.2.3 Présentation de la plate-forme	10
1.3 Présentation de l'équipe	11
2 Présentation du stage	13
2.1 Thème de stage : le CYCAB	13
2.1.1 Le concept	13
2.1.2 Caractéristiques	13
2.1.3 Fiche technique	14
2.2 Sujet du stage	16
II Etude théorique du problème	17
3 Génération et suivi de trajectoires en robotique terrestre	18
3.1 Quelques notions utiles	18
3.1.1 Clothoïde	18
3.1.2 Non holonomie	20
3.2 Génération de trajectoires pour véhicules à roues	20
3.3 Positionnement du véhicule lors du suivi de trajectoire	22
3.4 Chemins de Dubins pour véhicules non holonomes	22

III	Programation	24
4	Partie 1 : Génération de chemins de véhicules	25
4.1	Planification des chemins d'un carrefour	25
4.1.1	Approche	25
4.1.2	exemple de fonction de génération de chemin	26
4.1.3	Visualisation de tous les chemins générés	29
4.2	calcul du pas du maillage	30
4.3	Présentation du carrefour	30
5	Partie 2 : Contrôle de véhicules	32
5.1	Algorithme 1	32
5.2	Algorithme 2	34
6	Partie 3 : Reprise d'un simulateur existant et son adaptation pour la visualisation	37
6.1	Description du simulateur	37
6.2	Modèle utilisé	37
6.3	Modes de fonctionnement et d'affichage	38
6.3.1	La fenêtre principale	38
6.3.2	Fenêtre 2	38
6.3.3	Fichier.txt	39
6.4	Explication des programmes	39
6.5	Adaptation du simulateur	40
	Conclusion	44

Table des figures

1.1	Aperçu du laboratoire IMARA	10
1.2	Composition de l'équipe	12
2.1	Le CyCab	14
2.2	Dessin schématique du CyCab	15
3.1	Exemple de clothoïde	19
3.2	Présentation de la voiture	21
4.1	Visualisation de tous les chemins	29
4.2	Schéma pour le calcul du pas de discrétisation	30
4.3	Visualisation du carrefour	31
6.1	Premier affichage	41
6.2	Deuxième affichage	42
6.3	Affichage final	43

Remerciements

Je tiens à remercier vivement l'INRIA Rocquencourt de m'avoir accueilli pour effectuer mon stage de fin d'études dans le service IMARA, m'offrant la possibilité d'acquérir une expérience professionnelle très enrichissante.

Je remercie tout particulièrement monsieur Arnaud DE LA FORTELLE, mon responsable de stage, pour ses conseils et le temps qu'il a bien voulu me consacrer tout au long de ce stage. C'est grâce à lui et au travail proposé que ce stage a été formateur.

Je souhaite également présenter mes remerciements à toute l'équipe IMARA : Laurent BOURAOUI, Stéphane PETITT et Armand IVET pour leur aide lorsque certains problèmes se sont présentés.

Je remercie aussi tous les stagiaires présents sur place, dans le laboratoire IMARA, où se déroulait mon stage.

Introduction

Ce stage de fin d'études s'inscrit dans le cadre de mes études en Mathématiques Appliquées et Calcul Scientifique (M.A.C.S) à l'Institut Galilée, situé à l'Université Paris 13, pour l'obtention du diplôme d'ingénieur.

Il s'est déroulé à l'unité de recherche de **Rocquencourt** de l'Institut National de Recherche en Informatique et en Automatique (**I.N.R.I.A**).

L'un des buts majeurs de la robotique réside en la création de robots autonomes. Par robot, on entend un système mécanique équipé de capteurs, d'actionneurs et d'un système de contrôle permettant d'agir sur les actionneurs en fonction, d'une part, de la tâche à accomplir, et d'autre part, des informations fournies par les capteurs. Un robot est dit autonome si il est capable mener à bien une tâche sans intervention humaine.

Le développement d'un robot autonome pose de nombreux problèmes fondamentaux dans des domaines variés. À titre d'exemple, le problème de la perception de l'environnement par le robot (capteurs) est naturellement lié au domaine de la vision ; le problème du suivi de trajectoire est lié à la théorie du contrôle. Un autre problème fondamental est celui de la planification de trajectoires. En effet, un robot est amené à se déplacer. Le calcul de trajectoires doit donc être une tâche élémentaire pour un robot se voulant être autonome.

L'objectif de ce stage est de faire une simulation de trajectoire de véhicule, se déplaçant d'un point initial A à un point final B, en évitant les collisions et les obstacles. Pour cela, dans un premier temps, on générera les différents chemins possibles à l'approche de l'intersection de deux routes droites (aller tout droit, aller à gauche, aller à droite). Ensuite, on calculera le profil de vitesse de la voiture, en tenant compte des obstacles.

La première partie de ce rapport est consacrée à la présentation de l'INRIA et du stage : les domaines de recherche, le laboratoire dans lequel j'ai effectué mon stage. Dans la seconde partie, on introduit quelques notions nécessaires afin de bien poser le problème pour la génération de trajectoire

des véhicules. La dernière partie traite tout le travail effectué : génération de chemins de véhicules, contrôle du passage des véhicules à un carrefour, reprise d'un logiciel de simulation et sa modification pour visualiser les résultats obtenus.

Première partie

Présentation de l'entreprise et du stage

Chapitre 1

Présentation de l'INRIA

Créé en 1967 à Rocquencourt, l'INRIA est un établissement public à caractère scientifique et technique, placé sous la double tutelle du ministère de l'éducation nationale, de l'Enseignement supérieur et de la Recherche, et du ministère de l'Industrie, des Postes, et des Télécommunications.

1.1 Domaines de recherche

Les activités de l'INRIA comprennent la réalisation de systèmes expérimentaux, la recherche fondamentale et appliquée, le transfert de technologie, l'organisation d'échanges scientifiques internationaux et la diffusion des connaissances et du savoir-faire.

Ces activités associent informaticiens, automaticiens, mathématiciens dans le cadre de projets de recherche regroupés dans cinq grands thèmes, subdivisés en plusieurs axes :

1. Systèmes communicants

- Systèmes distribués et architectures réparties
- Réseaux et télécommunications
- Systèmes embarqués et mobilité
- Architecture et compilation

2. Systèmes cognitifs

- Modélisation statistique et apprentissage
- Images et vidéo : perception, indexation, communication
- Données multi-media : interprétation et interaction homme-machine

- Synthèse d'images et réalité virtuelle
- 3. **Systèmes symboliques**
 - Sécurité et fiabilité du logiciel
 - Structures algébriques et géométriques, algorithmes
 - Organisation des contenus et de la langue
- 4. **Systèmes numériques**
 - Automatique et systèmes complexes
 - Grilles et calcul haute-performance
 - Optimisation et problèmes inverses en stochastique ou en grande dimension
 - Modélisation, simulation et analyse numérique
- 5. **Systèmes biologiques**
 - Modélisation et simulation pour la biologie et la médecine

Depuis 1992, l'INRIA participe à de grands projets à finalité industrielle dans le cadre d'action de développement : programmes de durée déterminée (généralement trois à cinq ans), menés en partenariat avec des industriels et des usagers des technologies de l'information. Faisant participer plusieurs équipes de recherche, ces actions constituent des occasions de coordination, voire de fertilisations croisées entre projets de recherche et sociétés de technologie.

L'INRIA de Rocquencourt, situé près de Versailles, fait partie des six unités de recherche de l'INRIA.³⁹ équipes de recherche cohabitent sur le site, soit un total de 800 personnes travaillant sur le site. Parmi ces projets de recherche, on trouve le projet Informatique, Mathématiques et Automatique pour la Route Automatisée (**IMARA**).

1.2 Le projet IMARA

1.2.1 Domaine d'application

Le projet **IMARA** est destiné à coordonner et à transférer les efforts de recherche de l'INRIA qui peuvent être appliqués au domaine de la **route automatisée**. Ce projet inclut des recherches dans les domaines suivants :

- Le traitement du signal (filtrage, traitement de l'image,...)

- Le controle-commande du véhicule (accelaration, freinage, direction,...)
- les outils de programmetion temps réel distribué
- les communications
- la modélisation
- le controle et l'optimisation des systèmes de transport

1.2.2 Objectifs

Le projet a pour objectifs l'amilioration du transport routier en terme de sécurité, d'efficacité, de confort et de minimisation des nuisances. L'approche technique est centrée sur les aides à la conduite, pouvant aller jusqu'à une automatisation totale.

1.2.3 Présentation de la plate-forme

IMARA travaille sur différents véhicules et notamment sur le **CyCab**. Ce véhicule est une plate-forme flexible de développement et de test pour les algorithmes liés aux différentes problématiques étudiées par l'équipe. Les recherches actuelles concernent surtout la problématique de localisation, aspect essentiel pour permettre une navigation autonome du véhicule.



FIG. 1.1 – Aperçu du laboratoire IMARA

1.3 Présentation de l'équipe

Mon stage s'est déroulé au sein du service robotique dont le rôle est la mise en oeuvre des outils matériels et logiciels pour les expérimentations robotiques des projets de recherche du site.

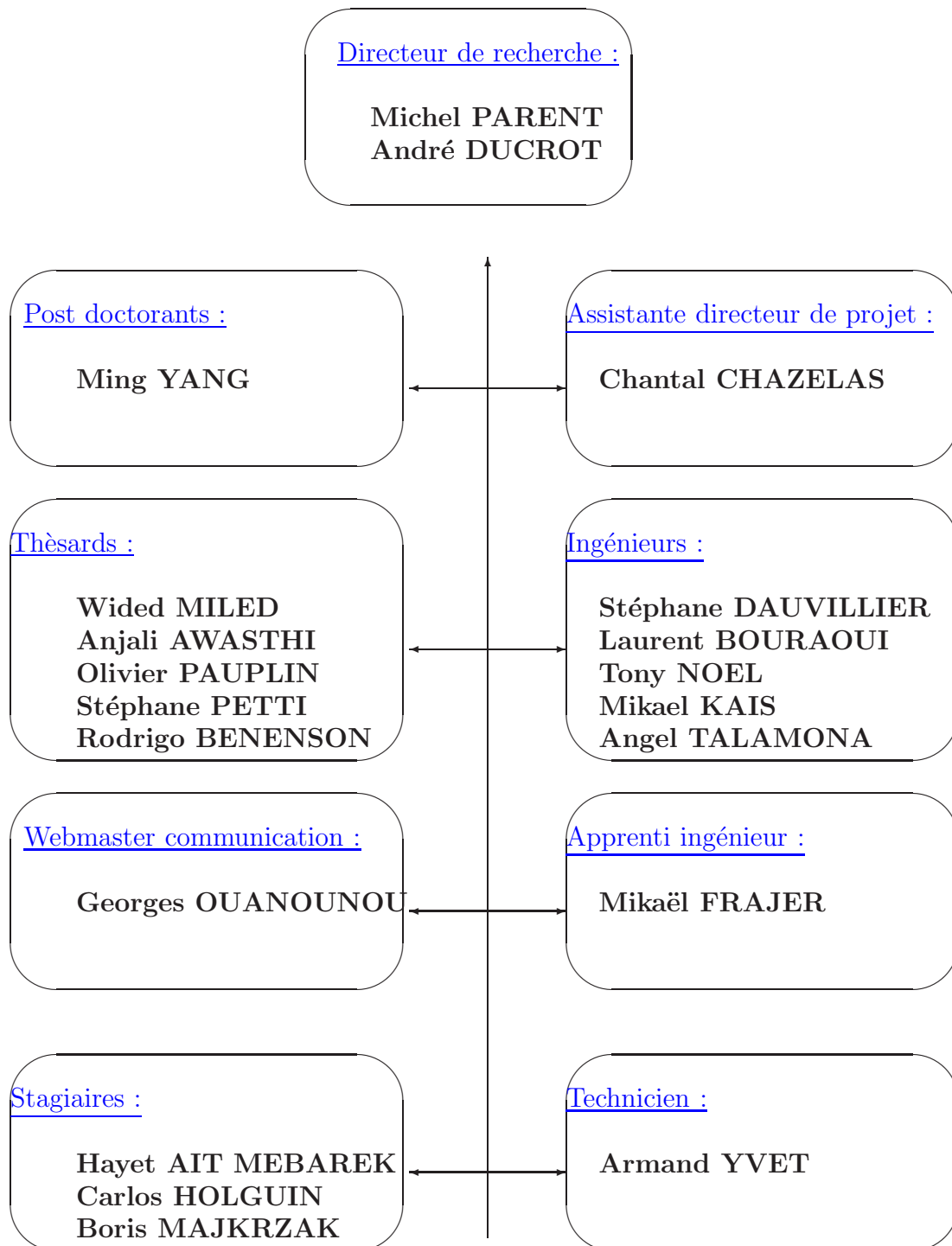


FIG. 1.2 – Composition de l'équipe

Chapitre 2

Présentation du stage

2.1 Thème de stage : le CYCAB

Dans le cadre de la route automatisée, l'INRIA a imaginé un système de transport original de véhicules de libre service pour la ville de demain. Ce système de transport public est basé sur une flotte de petits véhicules électriques spécifiquement conçus pour les zones où la circulation automobile doit être fortement restreinte. Pour illustrer ce système, un prototype, nommé le **Cycab**, a été réalisé.

2.1.1 Le concept

Le CyCab est une plate-forme qui a été développée par l'INRIA avec l'aide de l'INRETS, d'EDF, de la RATP et la société Andruet S.A dans le cadre de recherche sur des nouveaux moyens intelligents de transport urbain. Il est actuellement commercialisé par la société *Robosoft*.

L'objectif rejoint, bien sur, ceux de l'IMARA, c'est-à-dire offrir une alternative à la voiture particulière dans les centres urbains avec des véhicules non polluants, pratiques et doués d'une certaine autonomie de décision.

2.1.2 Caractéristiques

Le CyCab, dont dispose le laboratoire, peut transporter deux personnes à une vitesse théorique maximale de 28 km/h. Il dispose de :



FIG. 2.1 – Le CyCab

- 8 batteries
- 4 roues motrices et directrices
- 4 moteurs à courant continu de 1 kw

Il peut se conduire en mode automatique ou en mode manuel à l'aide de joystick.

L'architecture embarquée se compose de deux PC et une interface CAN.

- Le premier est équipé d'un système d'exploitation Linux/RTAI et de l'environnement de développement Syndex ; il est utilisé pour les développements bas niveau.
- Le second, sous Windows XP, prend en charge la couche applicative.

2.1.3 Fiche technique

- Longueur : 1,90 m
- Largeur : 1,20 m



FIG. 2.2 – Dessin schématique du CyCab

- Poids : 300 Kg
 - Motorisation : 4 moteurs électriques de 1 Kw
 - 4 roues motrices et directrices
 - Autonomie : 2 heures d'utilisation continue
 - Capacité d'accueil : deux adultes et deux enfants
 - Conduite automatique ou manuelle par joystick
1. Caméra CCD pour la téléopération
 2. Joystick central de commande pour la conduite sécurisée
 3. Terminal multimédia
 4. Caméra linéaire pour la conduite en train
 5. Balises infrarouges pour la conduite en train
 6. Capteurs ultrasons pour la détection d'obstacles
 7. Vérin de direction électrique
 8. Moteur électrique par roue
 9. Frein électrique par roue
 10. Batteries et gestionnaire automatique de charge
 11. Borne de recharge par induction fixée sur la voirie

2.2 Sujet du stage

Le sujet de mon stage s'intitule **Planification de trajectoire de véhicules autonomes**. Le but de ce stage est de faire une simulation de trajectoire de véhicule, se déplaçant d'un point initial A à un point final B, en évitant les collisions et les obstacles. Pour cela, dans une première partie, on générera les différents chemins possible à l'approche de l'intersection de deux routes droites (aller tout droit, aller à gauche, aller à droite). Dans une seconde partie, on calculera le profil de vitesse de la voiture, en tenant compte des obstacles.

Deuxième partie

Etude théorique du problème

Chapitre 3

Génération et suivi de trajectoires en robotique terrestre

La recherche sur la génération de trajectoires a, jusqu'au début des années 80, concerné les robots manipulateurs (bras articulés). À partir de cette date, la notion de génération de trajectoires pour véhicules a émergé et de nombreuses études ont permis de mieux situer le problème.

3.1 Quelques notions utiles

3.1.1 Clothoïde

Définition

Clothoïde vient du grec **klothain**, qui signifie **filer** (la laine). La forme de la courbe rappelle la forme du fil qui s'enroule autour du métier à tisser.

Elle est une courbe qui, parcourue à vitesse constante v , est telle que la courbure varie linéairement. L'angle de rotation du volant d'une voiture étant proportionnel à l'angle de braquage des roues, lui-même équivalent, au voisinage de zéro, à la courbure multipliée par la distance entre les deux roues avant et arrière de la voiture .

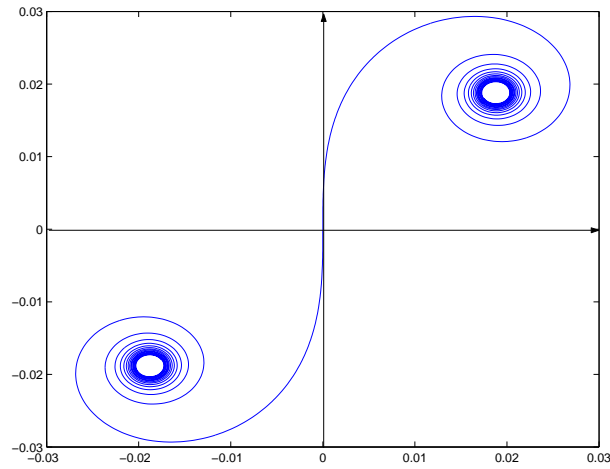


FIG. 3.1 – Exemple de clothoïde

Paramétrisation

$$\begin{cases} x(t) = a\sqrt{\pi}FresnelS\left(\frac{t}{\sqrt{\pi}}\right) \\ y(t) = a\sqrt{\pi}FresnelC\left(\frac{t}{\sqrt{\pi}}\right) \end{cases}$$

où :

$$\begin{cases} FresnelS(u) = \int_0^u \sin\left(\frac{\pi t^2}{2}\right) dt \\ FresnelC(u) = \int_0^u \cos\left(\frac{\pi t^2}{2}\right) dt \end{cases}$$

Propriétés

– Vitesse

$$v(t) = \begin{pmatrix} a \sin\left(\frac{t^2}{2}\right) \\ a \cos\left(\frac{t^2}{2}\right) \end{pmatrix}$$

– Accélération

$$a(t) = \begin{pmatrix} at \cos\left(\frac{t^2}{2}\right) \\ -at \sin\left(\frac{t^2}{2}\right) \end{pmatrix}$$

- Vecteur tangent

$$T(t) = \begin{pmatrix} \frac{a \sin\left(\frac{t^2}{2}\right)}{\sqrt{a^2}} \\ \frac{a \cos\left(\frac{t^2}{2}\right)}{\sqrt{a^2}} \end{pmatrix}$$

- longueur de l'arc

$$s(t) = \sqrt{a^2} t$$

3.1.2 Non holonomie

On dit qu'un véhicule est **non holonome** lorsque son mouvement est soumis à des contraintes s'exprimant par des relations non intégrables entre les composants du vecteur d'état et ses dérivées. Ainsi, une voiture ne peut se déplacer transversalement à sa direction.

3.2 Génération de trajectoires pour véhicules à roues

La difficulté de planification de trajectoires pour les robots terrestres sujets à des contraintes cinématiques (tels que les véhicules à roues) s'expliquent par le fait que pour une position donnée, il n'est pas possible, a priori, se mouvoir directement en tout point (non holonomie). C'est le cas pour les véhicules à roues de type voiture ou bicyclette où le **le modèle cinématique** s'exprime classiquement de la façon suivante :

$$\begin{cases} \dot{x} = \nu \cos(\theta) \\ \dot{y} = \nu \sin(\theta) \\ \dot{\theta} = -\frac{\nu \tan(\delta)}{L} \end{cases}$$

où :

- \mathbf{x} et \mathbf{y} sont les coordonnées du centre de l'essieu avant du véhicule (supposé non orientable)
- θ est l'angle que fait l'axe du véhicule avec l'axe des abscisses.
- δ est l'angle d'orientation de la roue arrière orientable par rapport à l'axe du véhicule.
- \mathbf{L} est la distance qui sépare la roue arrière de l'essieu avant.

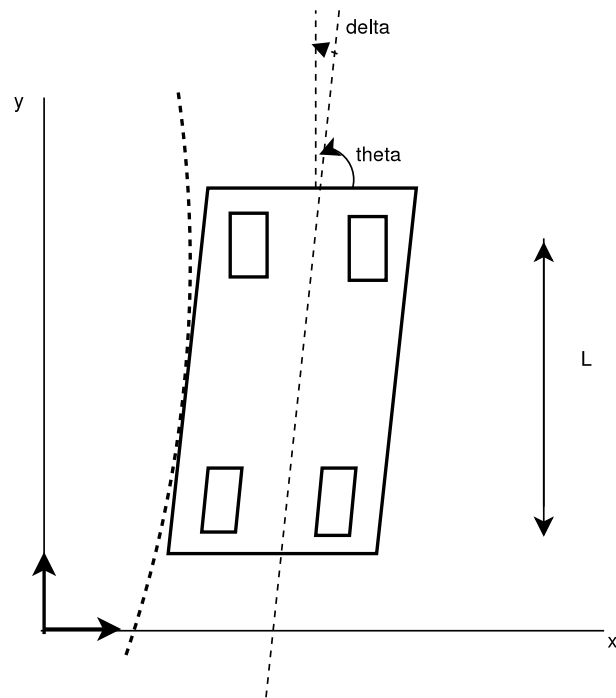


FIG. 3.2 – Présentation de la voiture

Le suivi de trajectoires planifiées en tenant compte de ces contraintes a été largement étudié en utilisant différentes techniques de commande et de positionnement en fonction du domaine d'application.

Lorsque le système d'équations différentielles exprimant les contraintes cinématiques peut être transformé en **forme chaînée**, on peut choisir des commandes sinusoïdales simples permettant de produire des mouvements qui modifient une composante d'état sans changer les autres. Des contraintes spécifiques liées à l'application ont conduit à développer d'autres types de commandes dérivées de cette première.

Dans le domaine de génération de trajectoires, la prise en compte de critères d'optimisation tel que le temps ou la consommation énergétique a donné lieu à de nombreuses recherches. On parle alors de **trajectoires optimales**.

Nous venons de voir que la non holonomie des véhicules à roues impliquait l'impossibilité, a priori, de se mouvoir directement en tout point de voisinage.

3.3 Positionnement du véhicule lors du suivi de trajectoire

Le suivi d'une trajectoire en robotique terrestre implique le positionnement du véhicule. Il peut se faire par **odomètres**, permettant d'obtenir un positionnement absolu.

3.4 Chemins de Dubins pour véhicules non holonomes

Dubins a prouvé que, pour certains véhicules non holonomes vérifiant le modèle cinématique évoqué précédemment, en l'absence d'obstacles, le plus court chemin entre deux configurations est une courbe obtenue par concaténation de segments de droites et d'arcs de cercle. Cette courbe est appelée **chemin de Dubins**

Les méthodes de construction de courbes de Dubins n'impliquent pas la continuité du rayon de courbure. Dans le cas où l'on souhaite assurer également cette continuité, les trajectoires planifiées sont formées d'un assemblage de segments de droite et de courbes appelées **clothoïdes**.

Troisième partie

Programation

Chapitre 4

Partie 1 : Génération de chemins de véhicules

4.1 Planification des chemins d'un carrefour

4.1.1 Approche

Cette partie consiste à planifier tous les chemins possibles que l'on peut avoir à un carrefour disposant d'une entrée et trois sorties(tourner à droite, tourner à gauche ou continuer tout droit).

On utilisera les courbes de cornu (clothoïdes) pour approcher les chemins de véhicules ainsi qu'une méthode d'approximation d'intégrales afin d'évaluer les fonctions de fresnels (j'ai utilisé la méthode de Gauss-Legendre qui est de l'ordre 3).

4.1.2 exemple de fonction de génération de chemin

```

function [T,X,Y,V,S,C]=chemin1(a,STR)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BUT      : générer un chemin de véhicule qui va tout droit
%           sur 50 mètres
% puis tourne à droite et avance sur 50 mètres
% DONNEES : -a : paramètre qui controle la courbure du véhicule
%           quand il tourne à droite
%           -STR : structure contenant quelques informations sur le
%           chemin
%           à suivre
% RESULTATS: T : vecteur de subdivisions du temps
%           X : coordonnées suivant x
%           Y : coordonnées suivant y
%           V : vecteur contenant la vitesse du véhicule à chaque pas
%           de temps
%           S : vecteur d'abscisses curvilignes
%           C : courbure du véhicule
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PARAMETRES
NBSUBD=300;\\
a=0.035;\\
% DEFINITION DES FONCTIONS DE FRESNEL
%Fresnels= @(t) sin(pi*(t.^2) /2);
%Fresnelc= @(t) cos(pi*(t.^2) /2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PREMIERE PARTIE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINITION DES VECTEURS-POSITION DE LA POSITION X ET Y SUR
%LA CLOTHOIDE
xpos=zeros(1,NBSUBD);
ypos=zeros(1,NBSUBD);

```

```

% PAS ENTRE 2 MESURES SUCCESSIVES=0.03m soit une vitesse de 3m/s
Y=[STR.ypoint11-50:0.03:STR.ypoint11];
DIM=length(Y);
X=STR.xbasdroit*ones(1,DIM);
% TIMERS =10 ms
%TEMPS
T=[0:0.01:0.01*(DIM-1)];
%ABSCISSE CURVILIGNE
S=Y;
%VITESSE
V=3*ones(1,DIM);
%COURBURE
C=zeros(1,DIM);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DEUXIEME PARTIE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X1=[];
Y1=[];
for i=1:NBSUBD
T1(i)=i/((sqrt(2)*NBSUBD));
S1(i)=T1(i)*a;
V1(i)=a;
C1(i)=T1(i)/a;

```

```

xpos(i)=a*sqrt(pi)*GausseLegendreApprox3
('Fresnels',0,i/((sqrt(2)*NBSUBD)/sqrt(pi)),100);
ypos(i)=a*sqrt(pi)*GausseLegendreApprox3
('Fresnelc',0,i/((sqrt(2)*NBSUBD)/sqrt(pi)),100);

X=[X,X(DIM+i-1)+xpos(i)];
Y=[Y,Y(DIM+i-1)+ypos(i)];
end
T=[T,T(end)+T1];
S=[S,S1];
V=[V,V1];
C=[C,C1];
DIM=length(X);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TROISIEME PARTIE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:NBSUBD
    X=[X,X(DIM+i-1)+ypos(NBSUBD-i+1)];
    Y=[Y,Y(DIM+i-1)+xpos(NBSUBD-i+1)];
    V=[V,V1(NBSUBD-i+1)];
    C=[C,C1(NBSUBD-i+1)];
    S=[S,S1(NBSUBD-i+1)];
    %T1=[T,T(end)+i/((sqrt(2)*NB_SUBD))];
end
T=[T,T(end)+T1];
DIM=length(X);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% QUATRIEME PARTIE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X1=[X(end):0.03:STR.xdroitbas];
Y1=ones(1,length(X1))*Y(end);
X=[X,X1];
Y=[Y,Y1];
T1=[0.01:0.01:0.01*(length(X1-1))];
T=[T,T(end)+T1];
%ABSCISSE CURVILIGNE
S=[S,zeros(1,length(X1));];
%VITESSE
V=[V,3*ones(1,length(X1))];
%COORBURE
C=[C,zeros(1,length(X1))];

```

4.1.3 Visualisation de tous les chemins générés

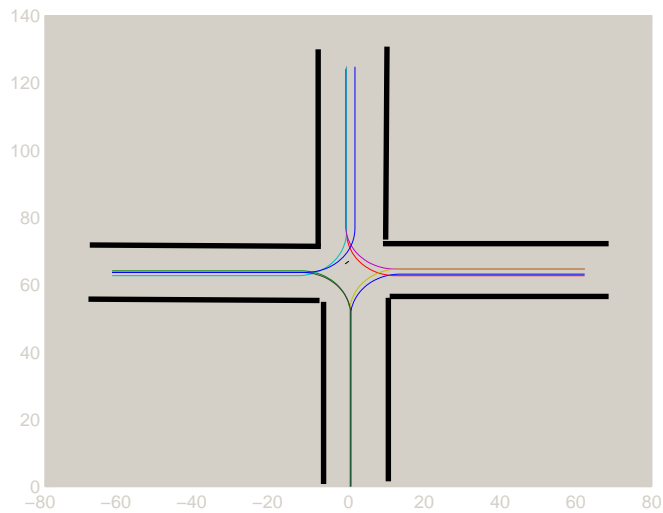


FIG. 4.1 – Visualisation de tous les chemins

4.2 calcul du pas du maillage

On souhaite imposer une certaine erreur raisonnable ϵ , aue l'on décide de fixer dès le départ (exemple : 0,06). Pour cela, on va calculer le pas de discrétisation de la trajectoire, sous forme de clothoïdes, afin de satisfaire cette contrainte.

D'une part, on peut déduire l'expression suivante :

FIG. 4.2 – Schéma pour le calcul du pas de discrétisation

$$\frac{d}{2} = R \sin \alpha \implies \cos \alpha = \sqrt{1 - \frac{d^2}{4R^2}}$$

D'autre part, si on impose une certaine erreur tolérée ϵ , on peut calculer le pas de maillage, noté \mathbf{d} , de la façon suivante :

$$\epsilon = R - R \cos \alpha \implies \epsilon = R \left(1 - \sqrt{1 - \left(1 - \frac{d^2}{4R^2} \right)} \right)$$

D'où :

$$d = 2R \sqrt{1 - \left(1 - \frac{\epsilon}{4 * R} \right)^2}$$

où :

- \mathbf{R} est le rayon de courbure
- \mathbf{d} est le pas nécessaire pour avoir une erreur raisonnable

4.3 Présentation du carrefour

On utilisera un carrefour à une entrée et 3 sorties comme présenté ci dessous.

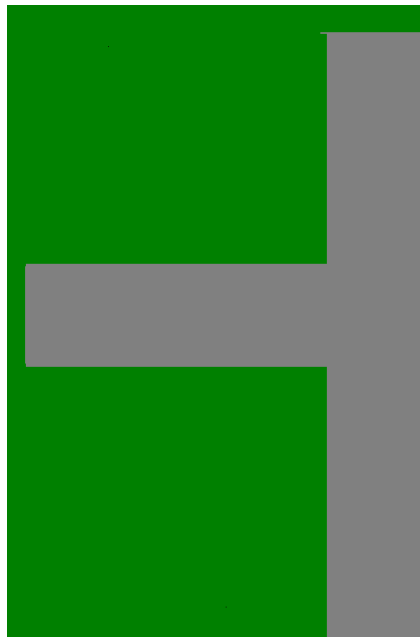


FIG. 4.3 – Visualisation du carrefour

Chapitre 5

Partie 2 : Contrôle de véhicules

Dans cette partie, on aimerait contrôler les trajectoires des véhicules afin d'éviter tout obstacle mobile. Pour cela, j'ai établi deux algorithmes qui contrôlent le croisement de deux véhicules dont on connaît les trajectoires à l'avance :

5.1 Algorithme 1

fonction [Voiture1,Voiture2] ← Algo1(Voiture1,Voiture2,DISTSECURE)

BUT :

faire une fonction qui calcule le profil de vitesse des deux véhicules : l'un vient de bas et tourne vers la droite l'autre vient de droite et continue tout droit Il accélérer le premier et ralentir le deuxième

DONNEES :

- **Voiture1** : structure contenant les informations sur la première voiture
- **Voiture2** : structure contenant les informations sur la deuxième voiture
- **DISTSECURE** : distance de sécurité à respecter (la distance nécessaire entre les centres de gravité des deux véhicules//

RESULTATS :

Les deux structures modifiées après calcul

a ← 0.035 ;

pour $ti \leftarrow 2 : \text{Voiture2.DIM}$

si $| \text{Voiture1.X}(ti) - \text{Voiture2.X}(ti) | \leq \text{DISTSECURE}$ et $| \text{Voiture1.Y}(ti) - \text{Voiture2.Y}(ti) | \leq \text{DISTSECURE}$

$k \leftarrow 1$;

si $\text{Voiture2.X}(ti) - \text{DISTSECURE} \leq \text{Voiture1.X}(ti-k)$ et $\text{Voiture2.Y}(ti) - \text{DISTSECURE} \leq \text{Voiture1.Y}(ti-k)$ et $0 \leq ti-k$

$k \leftarrow k+1$;

fin si

VOITURE 1

$\text{Voiture1.X}(ti) \leftarrow \text{Voiture2.X}(ti) - \text{DISTSECURE}$;

$\text{pas1} \leftarrow \frac{\text{Voiture1.X}(ti) - \text{Voiture1.X}(ti-k)}{k}$;

pour i allant de 1 à $k-1$

$\text{Voiture1.X}(ti-k+i) \leftarrow \text{Voiture1.X}(ti-k) + i \times \text{pas1}$;

fin pour

VOITURE 2

$t \leftarrow \frac{\text{Voiture2.T}(ti) + \text{Voiture2.T}(ti+1)}{2}$;

INSERTION DU POINT RAJOUTE POUR FREINER LE VÉHICULE 1

pour i allant de ti à $\text{Voiture2.DIM}-1$

$t \leftarrow \frac{\text{Voiture2.T}(i) + \text{Voiture2.T}(i+1)}{2}$;

si $\text{Voiture2.X}(i) - \text{Voiture2.X}(i+1) \neq 0$

$\text{Voiture2.X}(i) \leftarrow \text{Voiture2.X}(i) + a\sqrt{\pi} \times \text{GausseLegendreApprox3}('Fresnel', 0, \frac{t}{\sqrt{\pi}}, 100)$

si $\text{Voiture2.Y}(i) - \text{Voiture2.Y}(i+1) \neq 0$

$\text{Voiture2.Y}(i) \leftarrow \text{Voiture2.Y}(i) + a\sqrt{\pi} \times \text{GausseLegendreApprox3}('Fresnel', 0, \frac{t}{\sqrt{\pi}}, 100)$

fin si

fin si

si $\text{Voiture2.X}(i) - \text{Voiture2.X}(i+1) \neq 0$

```
    si Voiture2.Y(i)-Voiture2.Y(i+1)≠0
      Voiture2.Y(i) ← Voiture2.Y(i) +
        a√π×GausseLegendreApprox3('Fresnel', 0,  $\frac{t}{\sqrt{\pi}}$ , 100)
    fin si
  fin si
  Voiture2.T(ti)←t;
fin pour
fin si
fin pour
```

5.2 Algorithme 2

fonction [Voiture1,Voiture2] ← Algo1(Voiture1,Voiture2,DISTSECURE)

BUT :

faire une fonction qui calcule le profil de vitesse des deux véhicules : l'un vient de bas et tourne vers la droite l'autre vient de droite et continue tout droit Il faut maintenir le premier à sa vitesse initiale et decelerer le deuxieme

DONNEES :

- **Voiture1** : structure contenant les informations sur la premiere voiture
- **Voiture2** : structure contenant les informations sur la deuxieme voiture
- **DISTSECURE** : distance de securite à respecter (le distance nécessaire entre les centres de gravité des deux véhicules)

RESULTATS :

Les deux structures modifiées après calcul

```

function [Voiture1,Voiture2]=Algo2(Voiture1,Voiture2,DISTSECURE)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BUT      :faire une fonction qui calcule le profil de vitesse des deux
%           véhicules : l'un vient de bas et tourne vers la droite
%           l'autre vient de droite et continue tout droit
%           Il faut maintenir le premier à sa vitesse initiale
%           et decelerer le deuxieme
% DONNEES : -Voiture1: structure contenant les informations sur
%           la premiere voiture
%           -Voiture2: structure contenant les informations sur
%           la deuxieme voiture
%           -DIST_SECURE: distance de securite à respecter
%           (le distance nécessaire entre les centres
%           de gravité des deux véhicules
% RESULTATS: Les deux structures modifiées après calcul
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for ti=1:Voiture1.DIM
    tj=ti;
    TEST =0;
    while TEST ==0 & tj >= 1
        TEMP$_$DISTANCE=sqrt((Voiture1.X(tj)-Voiture2.X(ti))^2 +
                               (Voiture1.Y(tj)-Voiture2.Y(ti))^2);
        if TEMP$_$DISTANCE > DIST$_$SECURE
            TEST=1;
        else
            tj=tj-1;
        end
    end
end
end

```

```

if ti ~= tj
    % VOITURE 1
    % epsilon=DIST_SECURE-abs(Voiture1.X(ti)-Voiture2.X(ti));
    TEMP$_$DISTANCE=(Voiture1.X(ti)-Voiture2.X(ti))^2 +
                    (Voiture1.Y(ti)-Voiture2.Y(ti))^2;
    a=-2*(Voiture1.X(tj)-Voiture2.X(ti));
    b=DIST$_$SECURE^2-TEMP$_$DISTANCE;
    DELTA=a^2+4*b;
    epsilon$_$X1= (-a+sqrt(DELTA))/2 ;
    epsilon$_$X2= (-a-sqrt(DELTA))/2 ;
    epsilon$_$X=max(epsilon$_$X1,epsilon$_$X2);
    Voiture1.X(ti)=Voiture1.X(ti)-epsilon$_$X;
    %Voiture1.X(ti)=Voiture2.X(ti)-epsilon;
    pas1=(Voiture1.X(ti)-Voiture1.X(tj-1))/(ti-tj);

    for i=tj:ti-1
        Voiture1.X(i)=Voiture1.X(tj-1)+(i-tj+1)*pas1;
    end
end
end
end

```

Chapitre 6

Partie 3 : Reprise d'un simulateur existant et son adaptation pour la visualisation

Le simulateur utilisé est élaboré par un stagiaire de l'INRIA avec la collaboration de quelques membres de l'équipe IMARA. Il permet la localisation virtuelle d'un Cycab sur une carte routière.

6.1 Description du simulateur

Il s'agit de faire un simulateur du comportement du véhicule Cycab. le programme est codé à l'aide de l'éditeur Visual C++ de Microsoft. Ce programme est interfacé avec l'architecture du programme **Taxi** développé par les membres de l'équipe IMARA.

Le simulateur permet de visualiser la position du Cycab dans un espace 2D. Il s'agit donc de créer un programme à l'interface graphique dans lequel sera visible un plan (image bitmap) où on peut suivre le déplacement du Cycab virtuel.

6.2 Modèle utilisé

Le modèle utilisé étant **le modèle bicyclette**. Les équations qui régissent ce modèle sont simples. le véhicule doit être capable de connaître sa position à chaque instant. L'approche étudiée consiste à utiliser des **données odométriques**.

La localisation par **odomètre** utilise un codeur incrémental relatif, situé sur l'essieu arrière du véhicule et un codeur incrémental absolu sur la direction pour déterminer l'angle. A l'aide des données collectées périodiquement sur le déplacement de l'essieu et l'angle de braquage des roues, on peut construire la trajectoire du véhicule.

$$\begin{cases} x_{k+1} = x_k + \Delta_k \cos \theta_k \\ y_{k+1} = y_k + \Delta_k \sin \theta_k \end{cases}$$

où :

- Δ_k est la distance parcourue entre les instants k et $k + 1$.
- θ_k est l'angle de braquage des roues avant entre les instants k et $k + 1$.

Dans le cas du Cycab, les données sont relevées toutes les 10 ms.

6.3 Modes de fonctionnement et d'affichage

Le programme final est constitué de deux fenêtres et d'un fichier texte de sortie :

6.3.1 La fenêtre principale

D'une part, On affiche les valeurs envoyées au Cycab :

- la vitesse en mètre par seconde.
- l'angle au volant en radian.
- la distance totale parcourue depuis le début de la simulation.

D'autre part, on sélectionne la manière dont on veut piloter le Cycab virtuel :

- mode manuel : dans le cas où on dispose d'un fichier texte contenant la vitesse du Cycab et l'angle au volant, à chaque instant k .
- à l'aide d'un joystick connecté au PC.

6.3.2 Fenêtre 2

Grace à cette fenêtre, on peut suivre le déplacement du Cycab sur un plan 2D.

6.3.3 Fichier.txt

Il existe un module `recorder` qui permet d'enregistrer, dans un fichier texte, les variables utiles au système. On peut donc retracer les trajectoires de Cycab enregistrées lors des essais.

6.4 Explication des programmes

Comme on l'a dit précédemment, le programme est interfacé avec l'architecture du programme `Taxi`, développé par les membres de l'équipe IMARA. L'idée était de créer une classe `simulateur`, construite de la même manière que la classe `cycab` par un souci de clarté. La méthode a donc été de copier la classe `cycab` qui gère les paramètres liés au véhicule : envoi et réception des paramètres vitesse et angle ainsi que le contrôle de ces paramètres à l'aide de la fonction `core` qui représente le corps du cycab ; puis de la modifier pour permettre l'affichage du simulateur.

Le programme contient un fichier `main.cpp` contenant la fonction `WinMain()` nécessaire à l'exécution de toute API. Cette fonction principale, après avoir déclaré la classe `simulator` où sont déclarées toutes les autres classes, lance une seule fonction `startSim()`, qui permet le lancement du programme.

La fonction `startSim()` lance un thread principal qui permet l'ouverture de la fenêtre principale. Ensuite, en fonction de la sélection de l'entrée désirée (joystick ou fichier texte), un autre thread est lancé :

- `Thread_Proc_Joy` pour le joystick
- `Thread_Proc_File` pour le fichier

Ces deux threads sont similaires. Ils permettent déclaration, initialisation et ouverture de la deuxième fenêtre (et toutes les variables qui lui sont liées). Ces threads sont générés par des timers définis à 10 ms, c'est-à-dire que toutes les 10 ms, la commande est reçue puis envoyée à l'odomètre, qui la renvoie, à son tour, pour enfin calculer la position du véhicule, mettre à l'échelle de l'écran. Cette dernière tâche est réalisée grâce à la classe `Display` ; ce qui permet d'afficher le pixel à sa nouvelle position.

La classe `Mode` permet de contenir les variables à afficher ; en effet, selon le mode d'exécution choisi par l'utilisateur (joystick ou fichier texte), les variables lues sont recopiées dans cette classe, puis affichées dans la boîte de dialogue principale.

6.5 Adaptation du simulateur

L'un des objectifs de ce stage est de contrôler une flotte de véhicules, or, le simulateur utilisé est efficace seulement dans le cas d'un seul véhicule ; il fallait donc l'adapter pour visualiser plusieurs véhicules.

D'autre part, pour chaque véhicule, on a besoin d'un point de départ et un point d'arrivée, entre lesquels le véhicule se déplacera d'une manière raisonnable afin de respecter les vitesses autorisées ; or, le simulateur utilise toujours le même point de départ (initialisé dans les programmes une fois pour toutes) et calcule la trajectoire du véhicule à visualiser en fonction de l'angle au volant et de la vitesse à un pas de temps fixé. Il fallait donc modifier cela afin qu'un véhicule puisse se déplacer sans avoir une contrainte sur le point initial. Les entrées du programme ne sont plus l'angle au volant et la vitesse, mais les coordonnées des points traçant le chemin que le véhicule emprunte.

Voici, donc, le mode de fonctionnement de ce logiciel :
On lance le programme, deux fenêtres s'affichent(cf. fig)

- fenêtre affichant le carrefour
- fenêtre permettant de sélectionner le mode de fonctionnement(fichier texte ou à l'aide de joystick). Dans mon cas, le mode texte est approprié, car ma première partie de travail consistait à générer des chemins sous forme fichiers texte.

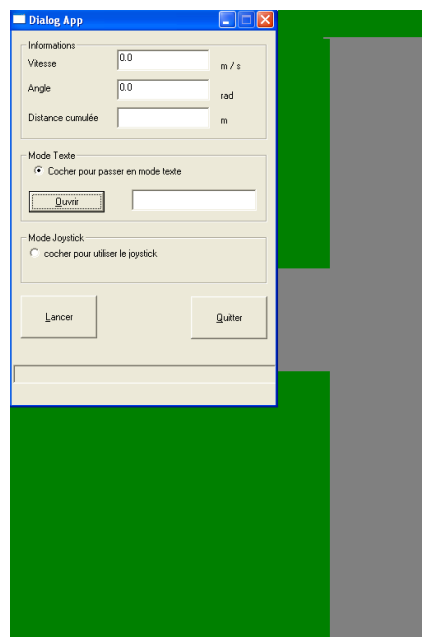


FIG. 6.1 – Premier affichage

On sélectionne donc le mode texte : une autre fenêtre s'affiche, permettant de choisir les fichiers textes dont on veut visualiser les chemins.

Après sélection des fichiers, la phase visualisation des chemins commence.

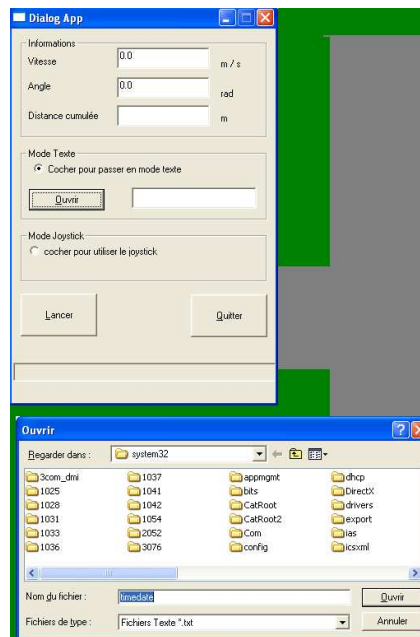


FIG. 6.2 – Deuxième affichage

En effet, au fur et à mesure de lecture des données des fichiers, des points s'afficheront, ce qui constitue le chemin de chaque véhicule(dans mon exemple, j'ai visualisé deux chemins).

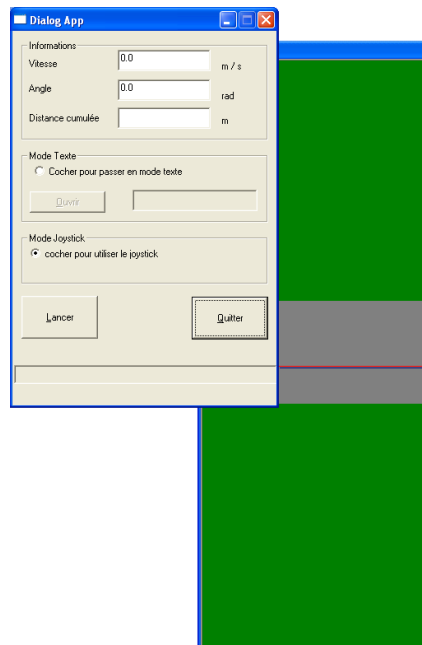


FIG. 6.3 – Affichage final

Conclusion

Dans ce taje, j'ai pu générer des chemins de véhicules, contrôler le passage de véhicules à un carrefour et visualiser ces résultats à l'aide d'un logiciel déjà établi, en le modifiant et l'adaptant à mes besoins

Ce stage m'a donné l'occasion de comprendre des programmes déjà existants, de les adapter afin de les utiliser dans mon travail. J'ai appris également beaucoup de notions que j'ignorais dans le domaine informatique : le stage m'a permis de renforcer mes connaissance en C++, par exemple.

La documentation technique laissée à l'INRIA sera reprise et complétée par un prochain stagiaire. Elle permettra l'utilisation et la poursuite de mon travail.