THÈSE

présentée par

Sofiane ABDOU

pour obtenir le grade de DOCTEUR
de l'**I**NSTITUT **N**ATIONAL **P**OLYTECHNIQUE DE **G**RENOBLE
(Arrêté ministériel du 30 mars 1992)

(Spécialité : Informatique, Système et Communication)

Spécification, Vérification et Implémentation de Missions Appliquées à des Véhicules Automatiques

---

Soutenue le 28 Octobre 1997 devant le jury composé de :

Messieurs	**Augustin**	**Lux**	Président
	Patrick	**Rives**	Rapporteurs
	Claude	**Laurgeau**	
	Bernard	**Espiau**	Examinateurs
	Michel	**Parent**	

Thèse préparée dans l'action de développement PRAXIT ÈLE à l'INRIA Rocquencourt et en collaboration avec le projet BIP de l'INRIA Rhône-Alpes.

A ma famille, avec une
pensée particulière pour ma mère.

Remerciements

Je voudrais remercier

*Monsieur Augustin Lux, Professeur à l'Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (*ENSIMAG*), qui a bien voulu présider mon jury ;*

Monsieur Patrick Rives, Directeur de recherche INRIA*, et Monsieur Claude Laurgeau, Professeur et Directeur du centre de robotique (*CAOR*) à l'Ecole Nationale Supérieure des Mines de Paris (*ENSMP*), qui se sont acquittés de la lourde tâche de rappoteurs ;*

Monsieur Michel Parent, Directeur de Recherche INRIA et responsable du projet PRAXITÈLE*, qui m'a accueilli dans son équipe et m'avoir permis de réaliser ce travail dans les meilleures conditions ;*

Monsieur Bernard Espiau, Directeur de Recherche INRIA *et responsable du projet BIP, pour avoir accépté d'encadrer mon travail, et qui m'a guidé tout en me laissant la plus grande liberté.*

tous les amis pour leur soutien sans faille.

ments, sont qualifiés de systèmes hybrides. Pour une programmation facile et sûre de la mission, une méthodologie de spécification systématique, ainsi que sa validation pour son implementation sur une architecture cible sont nécessaires. A chaque composante du système robotique, deux théories doivent être considérées : la théorie de l'automatique pour la conception des lois de commande et la théorie des systèmes réactifs pour les aspects discrets. Les langages synchrones, qui ont un formalisme permettant de décrire l'ordre et les relations entre les signaux d'entrées et de sorties correspondantes du système réactif, ont apporté une grande contribution dans ce domaine. Les compilateurs de ces langages peuvent produire des modèles, le plus souvent sous forme de systèmes de transition. Ces modèles peuvent être utilisés pour vérifier les relations entre les signaux d'entrées et de sorties du système (propriétés statiques). Nous décrirons l'utilisation d'une approche structurée de spécification et de vérification sur des véhicules automatiques à travers une mission du projet PRAXITÈLE. Le but de la mission est la distribution de véhicules vides par un seul conducteur. Des lois de commandes longitudinale et latérale sont établies pour réaliser un " train virtuel" de véhicules sans lien matériel. Quand le véhicule est arrivé à sa destination, il quitte le train pour se garer tout seul. Un générateur de trajectoire est alors nécessaire pour réaliser les manœvres de parking. L'environnement utilisé, appelé ORCCAD, propose une approche cohérente de la spécification haut niveau des missions jusqu'à leurs implémentations, en combinant événements discrets et aspects continus. Des fonctionnalités, basées sur une modélisation générique des robots mobiles à roues, sont intégrées dans cet environement pour faciliter la conception des lois de commande.

Mots clés : Robots Mobiles à Roues, Langages Synchrones, ESTEREL, ORCCAD, Systèmes de Contrôle de Véhicules Autonomes, Temps Réel, Génération de Trajectoire, Environnement de Programmation, Vérification Formelle.

Abstract: In this thesis, we descript how to program complex missions in robotic systems. Robotic systems are hybrid systems operating in real-time, handling events as well as "continuous computations". In order to program them easily and reliably, a systematic method for the specification of the mission, its validation and its efficient implementation on a target architecture are required. For every component of a robotic system, two well-known theories should be considered : the "automatic control" theory for the design of control laws and the "reactive systems" theory for the discrete events aspects. The family of synchronous languages, which aim to describe the complex ordering and causality relations between the inputs and the corresponding outputs of a reactive system, has been a very important contribution to the domain. These languages may be compiled into a wide class of models, usually labeled transition systems. The models are then used to perform static verification of *properties*, i.e order relations between inputs and outputs of the system. We describe the use of structural specification and formal verification in experiments involving the PRAXITÈLE project's automated vehicles. The goal of the programmed mission is to redistribute a number of empty vehicles, driving by only one driver. Longitudinal and lateral controllers are required to realize the "virtual train" without hard link. When a vehicle has reached its destination, it can break away from the platoon and parks by itself. A path planning is then required to realize parking manoeuvres. The environment used, called ORCCAD, proposes a coherent approach from high-level specification of a mission down to its implementation, by harmonious integration of discrete and continuous aspects. A set of fonctions, based on the generic modelisation of wheeled mobile robots, are integrated into this environment in order to facilitate the design of the control laws.

Keywords: Wheeled Mobile Robots, Synchronous Languages, ESTEREL, ORCCAD, Automated Vehicle Control Systems, Real Time, Path Planning, Programming Environnement, Formal Verification.

Table des matières

Table des figures

Introduction Générale

UN robot autonome doit effectuer des tâches non répétitives dans un environnement inconnu. Dans ce contexte, la spécification de la mission doit permettre au robot d'atteindre l'objectif en *réagissant* à des événements par exécution des *actions* appropriées. La théorie de l'*automatique* permet de réaliser les actions physiques et le programme qui les enchaînent doit les rendre *réactives* vis à vis de ces événements. Afin de satisfaire cette double exigence, l'architecture de contrôle, en liaison avec les outils logiciels, doit offrir une approche de spécification cohérente, verticale et accessible sur plusieurs niveaux. La première couche est destinée à l'*ingénieur* chargé de mettre en œuvre le système en vue d'une mission donnée ; la deuxième est accessible à l'*automaticien*, concepteur des actions élémentaires, qui met au point les lois d'asservissement et l'environnement événementiel associé (conditions de démarrage et d'arrêt, exceptions). Son travail de conception est facilité si les fonctionnalités de base lui sont fournies. Ces fonctionnalités, qui sont très liées à l'architecture-cible et au type d'applications visées, sont mises à sa disposition par l'*informaticien-électronicien* spécialiste du temps réel.

Depuis les armoires à relais, la logique séquentielle et la logique câblée, les concepteurs des systèmes robotiques ont vu peu à peu évoluer leur métier avec l'apparition de la logique programmée et à travers la révolution des micro-processeurs. La plupart d'entre eux ont ainsi été contraints d'abandonner leur mode de pensée hérité de la culture des électroniciens pour adopter celle des informaticiens qui croyaient leur fournir des outils et des langages. Des langages de spécifications textuelles sont apparus et commencent à laisser peu à peu la place à des langages de spécification graphiques plus proches de la culture du concepteur. Ce dernier, et quel que soit son niveau d'intervention dans la hiérarchie de l'architecture, n'a pas seulement le souci de programmer, mais aussi celui de *spécifier*, de *valider*, puis d'*implémenter*, avant de prouver finalement que ce qu'il a réalisé est sans faute.

Le problème est que les langages informatiques proposés étaient inadaptés à leurs besoins. En effet, le concept séquentiel venait se heurter au parallélisme intrinsèque aux problèmes traités. Et le parallèlisme d'exécution des informaticiens ne répondait pas au problème du parallélisme d'expression des concepteurs. Leur premier besoin reste donc un formalisme adapté à leur culture qui leur permette de garantir la correction de ce qu'ils expriment, et par conséquent de découvrir les erreurs de conception au plus tôt.

Au niveau de la conception des lois de commandes, l'automaticien veut :

- spécifier suivant une organisation hiérarchique ;
- réutiliser plutôt que de réécrire ;
- détecter les incohérences éventuelles au plus tôt ;
- gagner en fiabilité et sécurité du logiciel par des moyens simples : code produit automatiquement et lisible ; réutilisation du code validé et vérification formelle.

L'ingénieur spécifie une mission, en utilisant les lois de commandes déjà écrites. A ce niveau, la spécification devrait :

- permettre d'augmenter l'*autonomie* dans la réalisation des actions robotiques réflexes ;
- garantir la fiabilité. Pour cela, il faut vérifier autant que possible le bon fonctionnement de la mission. Ceci inclut la validation du comportement spécifié.

Le système ORCCAD constitue une réponse satisfaisante aux besoins mentionnés plus haut. L'approche du système définit une mission sous la forme d'actions robotiques, soit élémentaires soit complexes, afin d'accomplir l'objectif de la mission. L'action élémentaire robotique est représentée par l'entité **T***âche*-**R***obot* (TR) qui est un programme multi-tâches qui réunit des aspects algorithmiques et logiques. La **P***rocédure*-**R***obot* (PR) est une action complexe représentée par un arrangement logique des actions robotiques élémentaires. Finalement une **T***âche*-**M***odule* (TM) est une tâche temps-réel, une TR est un ensemble de TM communicantes et une PR est constituée d'une combinaison de TR et de PR.

La spécification avec ORCCAD

Un modèle est associé à chacun des deux niveaux de spécification des actions robotiques :

- Le modèle flot de données est utilisé pour décrire les lois de commande sous forme de bloc-diagrammes, si chers aux automaticiens. Ce niveau est réalisé avec l'utilisation des primitives de l'OS temps réel VXWORKS ;
- Le modèle flot de contrôle est utilisé pour spécifier des missions où plusieurs actions sont enchaînées. A ce niveau la *réactivité* des actions est une caractéristique fondamentale. Son traitement doit être fait dans un cadre formel afin de pouvoir vérifier la correction de son comportement. Différents langages de spécification haut niveau existent. Rares ceux qui sont vérifiables à partir de modèle représentant - de façon plus au moins abstraite - l'ensemble des comportements possibles du programme. Nous utiliserons l'approche *synchrone* qui permet d'offrir un formalisme de haut niveau pour décrire les modèles. Et pour la vérification des propriétés, nous utiliserons une technique de vérification formelle basée sur la réduction de la taille du modèle.

Objet de la thèse

Notre travail entre dans le cadre du projet PRAXITÈLE qui consiste à *automatiser* des véhicules électriques pour les utiliser dans le transport. Notre objectif est d'adapter et d'utiliser les outils du système ORCCAD pour spécifier, vérifier et implémenter des missions complexes sur ces véhicules.

Contribution

Les apports fondamentaux de notre travail, tels qu'ils sont décrits dans le mémoire concernent principalement :

L'adaptation d'ORCCAD aux véhicules automatiques

L'adaptation nécessite :

- L'intégration des fonctionnalités de contrôle-commande de base, dans l'environnement ORCCAD pour le véhicule ATALANTE[1]. Cette intégration doit aborder d'une manière générique le problème de la modélisation cinématique des robots mobiles à roues ;

- Validation et tests de ces fonctionnalités en les utilisant dans une application. Cette étape consiste à spécifier la plus grande entité que l'automaticien peut concevoir : la TR. Le code est généré automatiquement par ORCCAD à partir de la description des commandes et à l'aide des fonctionnalités.

La spécification, la vérification et l'implémentation de missions

Plusieurs missions sont visées par PRAXITÈLE avec une complexité croissante. Un gain en productivité et en fiabilité peut être obtenu en réutilisant les fonctionnalités de base pour écrire les TRs. La spécification en PRs combinant un ensemble de lois de commande permet d'augmenter l'autonomie du robot par réactivité. En effet, en restant constamment en interaction avec son environnement, le robot réagit à des événements en lançant les TRs spécifiées.

Plusieurs PRs ont été spécifiées, vérifiées et implémentées :

- la mission de ramassage de véhicules sous forme d'un train virtuel utilisant le formalisme des PRs ;

- une mission de parking parallèle ;

- enfin, ces deux missions sont enchaînées pour réaliser une application plus complexe ;

1. le véhicule automatique que nous utiliserons pour nos expérimentations.

Organisation du document

Le document est organisé en deux parties :

– La première partie expose les méthodes et les outils de programmation d'un robot autonome ;

– La deuxième partie est dédiée aux mises en œuvres réalisées.

Le contenu des deux parties est le suivant :

Première partie : Les méthodes et les outils de programmation

- Architectures et Environnements de Programmation : Ce chapitre présente l'état de l'art des architectures de contrôle permettant de donner aux robots des capacités de raisonnement et d'actions. Plusieurs systèmes seront présentés et les services apportés par les outils logiciels pour faire fonctionner une architecture seront illustrés à travers des exemples.

- Outils de Spécification et de Vérification des Systèmes Réactifs : La réactivité est une caractéristique fondamentale des systèmes robotiques et la sécurité de leur fonctionnement causée par une erreur de conception peut être coûteuse. Dans ce chapitre, nous présenterons les méthodes de spécification de ces systèmes ainsi que les méthodes de vérification et les outils associés.

- ORCCAD : Un Environnement de Programmation Robotique de Nature Hybride : Ce chapitre présente ORCCAD qui est un environnement de programmation de nature hybride utilisant l'approche synchrone pour la spécification. Une classification générique des entités de base de cet environnement (les **T***âche*-**M***odules*) pour les robots mobiles à roues a été proposée. Cette classification permet de bien structurer et d'adapter ORCCAD pour ce type de robot.

Deuxième partie : Les mises en œuvre

- Automatisation des Transports : Ce chapitre est une introduction générale présentant les projets semblables à PRAXITÈLE. Après la présentation des différentes versions de PRAXITÈLE, je me suis situé dans le projet, notamment sur les versions 2 et 3 où l'automatisation prend un aspect plus important. Un état de l'art des travaux actuels couvrant l'automatisation des véhicules et les commandes longitudinale et latérale a été proposé.

- Fonctionnalités de Base pour le Contrôle-Commande de Véhicules : Des fonctionnalités comme des structures de commande prédéfinies ou des modèles de robots mobiles sont à la disposition de l'automaticien pour faciliter la conception des actions élémentaires. Dans ce chapitre, nous présenterons toutes les fonctionnalités -traitement des capteurs- pour le contrôle-commande d'ATALANTE.

- Le Train de Véhicules : En utilisant les fonctionnalités de base, les asservissements des moteurs sont réalisés. Un train de véhicules utilisant un capteur de position et les asservissements est réalisé avec l'approche de programmation d'ORCCAD. Une première spécification est faite en utilisant une TR et une deuxième avec le formalisme des PRs ;

- Manoeuvre de Parking Parallèle : Un scénario de mise en stationnement automatique est spécifié, vérifié et implémenté avec le formalisme des PRs. Le scénario est ensuite enchaîné avec l'application de train de véhicules pour réaliser une mission complexe.

Première partie

Les Outils et les Méthodes

Chapitre 1

Architectures et Environnements de Programmation

DEPUIS maintenant plus de deux décennies on cherche à concevoir des contrôleurs permettant aux robots d'accomplir des missions complexes, dans des environnements quelconques (non adaptés) et avec le minimum d'intervention. Avant de faire un choix d'un contrôleur, il nous parait évident que l'évaluation de ses performances est une issue qui peut être déterminante. Cependant, comme nous allons le voir dans de ce chapitre, les attributs qui dimensionnent un contrôleur autonome sont nombreux et difficiles à estimer.

Pour trouver l'architecture de contrôle qui permet de programmer des missions robotiques, nous nous inspirons de la littérature. Une classification des diverses approches sera faite, des critiques seront proposées et des conclusions seront tirées pour établir l'architecture la mieux adapter à nos besoins.

D'autre part, si les architectures privilégient l'analyse des besoins et des fonctionnalités pour attribuer aux robots des capacités de raisonnement et d'actions, ils n'ont pas de méthodologie générale, ni d'outils pour structurer et intégrer les programmes. En partant des besoins de fonctionnement d'une architecture, nous présenterons plusieurs environnements qui ont pu se traduire par des intégrations sur différents robots.

Le plan suivi dans ce chapitre est le suivant :

- ☞ dans la section 1.1 les principaux critères qui dimensionnent les performances d'un contrôleur robotique autonome seront donnés ;
- ☞ la section 1.2 présente l'état de l'art des architectures de contrôle. Une classification par rapport à des critères tels que la réactivité, la décomposition hiérarchique sera illustrée à travers plusieurs systèmes ;
- ☞ la section 1.3 introduit les architectures hybrides. Ces dernières ont été développées pour compenser les points faibles des architectures traditionnelles. Une comparaison de plusieurs systèmes, basée sur la place occupée par les composantes du système dans la hiérarchie hybride, sera faite ;

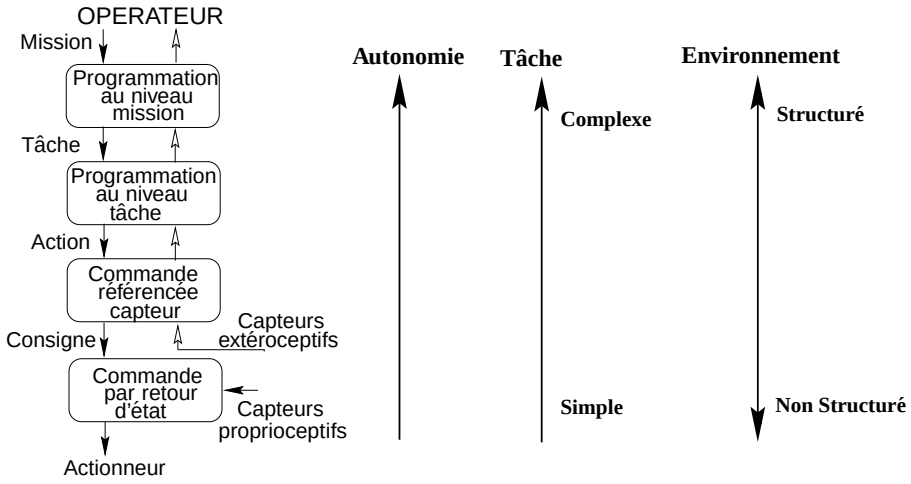


FIG. 1.1-1 – *Niveaux d'Autonomie.*

- ☞ nous présentons dans la section 1.4 les services attendus par un environnement de programmation sous lequel une architecture doit fonctionner ;
- ☞ des exemples d'environnements les plus représentatifs sont donnés dans la section 1.5.

1.1 Critères de dimensionnement d'un contrôleur autonome

Un contrôleur *autonome* doit avoir l'énergie et la capacité de s'*auto-gouverner* c'est à dire qu'il doit être capable d'exécuter plusieurs fonctions sans intervention extérieure. Plusieurs nouvelles fonctions comme : la réponse réflexe, le traitement des exceptions, la modélisation de l'environnement, le raisonnement et la prise de décisions, la tolérance à la défaillance...seront ajoutées à la fonction de contrôle conventionnelle pour la conception de l'espace d'intelligence d'un robot autonome. Cet espace est dimensionné par plusieurs critères [DK90]. Les plus importantes sont le **niveau de l'autonomie** et la **complexité de la mission**. Lors de la définition des missions d'un robot, des compromis doivent être faits entre ces deux critères pour maintenir l'intelligence du robot à un niveau raisonnable.

1.1.1 Le niveau de l'autonomie

Le niveau de l'autonomie peut être défini comme la probabilité qu'une action inattendue soit engagée par le robot sans l'intervention d'un opérateur. Le niveau de l'autonomie correspond à l'échelle de la figure 1.1-1. Au niveau le plus bas de l'autonomie se trouve les systèmes directement contrôlés par un opérateur humain. Les niveaux au dessus devront demander moins d'interventions. Pour faire évoluer le système robotique, des méthodes classiques d'automatique sont utilisées sur les "niveaux bas", et des techniques algorithmiques de parcourt d'arbre, d'optimisation, de décomposition, d'intelligence artificielle...sur les "niveaux hauts".

Niveau commande par retour d'état

L'opérateur donne une consigne. L'exécution de l'asservissement aura besoin de l'état interne du système. Le suivi d'une trajectoire pré-définie en est un exemple typique.

Niveau commande référencée capteur

Le système perçoit son environnement, à l'aide de capteurs extéroceptifs, avant de déduire la consigne et de passer à l'action d'asservissement. Cette capacité lui permet d'être en contact permanent avec les changements qui peuvent survenir dans l'environnement. Par exemple, le suivi de trajectoire avec évitement d'obstacle est une application qui exécute continuellement la boucle perception/action en s'asservissant relativement à l'environnement.

Niveau programmation de tâche

A ce niveau, le robot a une autonomie dans la sélection des actions, en plus de la capacité de les adapter dans l'environnement. Connaissant a *priori* l'ensemble des différentes situations possibles, le système sélectionne les actions appropriées pour arriver au but fixé par la tâche. Ainsi, le robot peut atteindre le but mais de différentes façons en fonction de sa situation dans l'environnement et dans le temps. Par exemple, la tâche "gares_toi_à_cette_place", nécessite la localisation de la place, puis le calcul des trajectoires avant de les exécuter en évitant les obstacles. Pour deux exécutions consécutives de cette tâche, les actions ne seront pas toujours identiques. Ils vont dépendre des aléas de l'environnement.

Niveau programmation de mission

Une planification des tâches est nécessaire pour charger le robot d'une mission dans un environnement non connu a priori. Ce niveau représente l'autonomie totale sans intervention où le robot doit être doté d'une "intelligence" pour atteindre les objectifs fixés par la mission. Par exemple dans le cadre d'une mission "Aller au bâtiment X et se garer au maximum à Y métres", son exécution nécessite une stratégie pour trouver le chemin le moins coûteux en énergie, la place où la voiture pourra se garer plus facilement.

1.1.2 La complexité de la mission

La difficulté de dimensionner une mission réside dans le fait que son spectre est trop large pour être représenté par un seul critaire. Cela dépend des extensions envisagées et surtout de **la complexité de l'environnement** et **la complexité des tâches**.

L'environnement est peu complexe quand il est statique, c'est à dire qu'il n'y a pas de changement dans l'environnement autre que ceux que l'on a planifié. Un environnement bien structuré est facilement modélisable et sa perception est fiable. Sa structuration peut se faire par introduction de marques de navigation par exemple, ceci devrait faciliter la mission et la rendre indépendante de l'environnement.

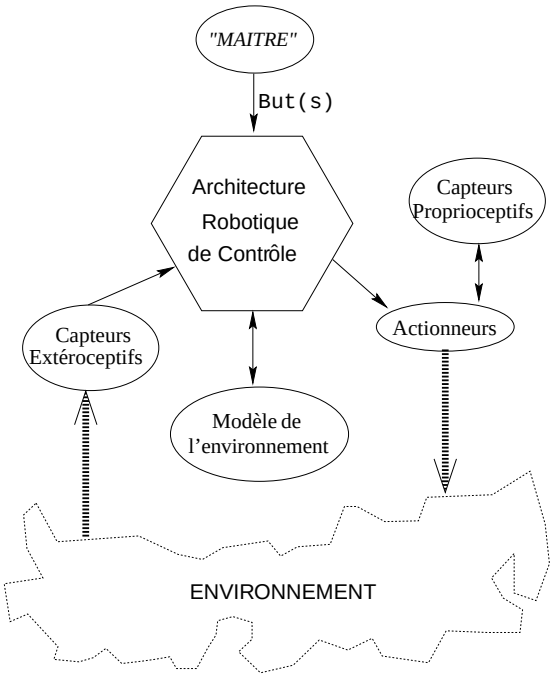


FIG. 1.1-2 – *Architecture simplifiée d'un contrôleur robotique.*

Les tâches sont dites simples quand le système a très peu de choix entre les différents chemins pour atteindre le but. L'assemblage des voitures dans une usine est un exemple de tâches simples, puisque les robots font tout le temps la même séquence d'actions. Ce n'est pas étonnant, puisque cet environnement reste tout le temps très simple et prédictible, donc peu complexe.

Pour diminuer ce critère, les tâches estimées difficiles dans la mission, sont affectées à un opérateur humain.

1.1.3 Conclusion

On dira qu'un robot X est plus autonome qu'un robot Y dans deux cas : si dans un même environnement le robot X peut accomplir une mission "jugée" plus complexe que celle du robot Y ; ou bien si les robots X et Y peuvent accomplir la même mission dans un environnement "a priori" plus complexe pour le premier robot. Cependant, tout le problème est de mesurer la complexité de la mission et de l'environnement.

1.2 Les approches dans les architectures de contrôles

Les architectures de contrôles sont des systèmes sophistiqués permettant aux robots de faire un travail physique utile dans un environnement réel. Le système de contrôle perçoit continuellement son environnement grâce à des capteurs extéroceptifs, il met à jour le modèle interne de l'environnement, et passe les consignes à l'actionneur pour atteindre un but assigné par un "maître" (figure 1.1-2). Pour une réalisation efficace et organisée, plusieurs

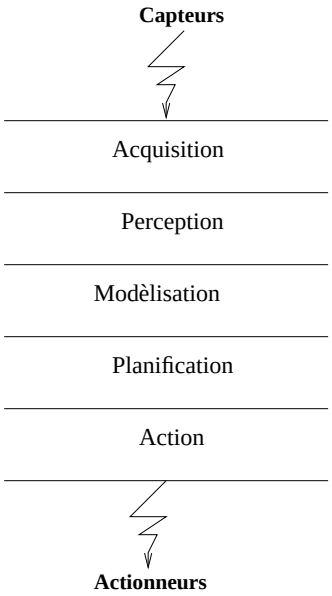


FIG. 1.2-3 – *Une architecture hiérarchique d'un robot autonome.*

approches ont été proposées. Dans la suite, une classification des diverses approches est illustrée à travers quelques exemples.

1.2.1 Architectures hiérarchiques

C'est l'approche la plus naturelle qui repose sur la décomposition fonctionnelle apparaissant dans la figure 1.2-3. Elle consiste à décomposer les tâches en sous-tâches d'un niveau de l'hiérarchie. Le flot de contrôle permet d'initialiser, de démarrer et de terminer les sous-tâches. Par contre, le flot de données consiste à fusionner, assimiler les données capteurs et modéliser l'environnement. Un robot doit commencer par percevoir l'environnement en recueillant les données par des capteurs. Il lui faudra ensuite analyser les données et modéliser son environnement pour pouvoir planifier les tâches. La partie Modélisation-Planification (dite aussi délibération) peut être forte dans le contrôle d'un système expert ou totalement négligée comme un simple thermostat dans un système de contrôle mécanique. L'action est directement liée aux données captées, mais elle dépend aussi de leur interprétation. L'interprétation peut se traduire par la proposition d'une action toute simple (une consigne), comme elle peut proposer un plan d'actions compliqué avec des conditions non linéaires.

Exemples d'architectures hiérarchiques

L'architecture NASREM proposée par Albus [AML89] est une référence en terme d'architectures hiérarchiques. L'architecture comporte six couches (figure 1.2-4) qui sont décomposées horizontalement en trois parties. Les termes de la décomposition verticale sont l'augmentation du degré d'abstraction et la diminution de la fréquence d'interaction et des

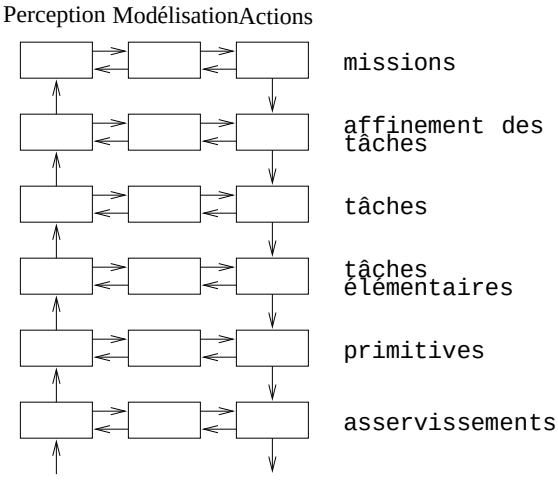


FIG. 1.2-4 – *L'architecture hiérarchique* NASREM.

dynamiques avec l'environnement. La décomposition horizontale correspond à la réactivité.

Aux niveaux les plus bas se trouvent les traitements rapides des données tels que les asservissements. Aux niveaux supérieurs se trouvent les opérations qui demandent plus de raisonnement sur les données. Les couches de NASREM ont été utilisées dans plusieurs systèmes : télé-robotique spatiale, de manœuvre et d'autonomie d'un sous-marin, de conduite à distance [Alb95].

L'architecture AURA **(Autonomous Robot Architecture)** a été développée par Arkin [Ark87]. Elle est composée de cinq couches : un planificateur de mission qui délivre un ensemble de buts à atteindre, un module de navigation qui à partir d'une carte construit un chemin permettant de satisfaire les buts, et un pilote basé sur le contrôle homéostatique. Ce dernier assure le déplacement du robot selon le chemin fourni. La fonction du contrôle homéostatique (prises en compte d'informations proprioceptives) est basée sur l'observation des aspects critiques tels que le niveau d'énergie, l'équilibre du véhicule.

Remarques 1.2.1

1. *Du point de vue de la perception de l'environnement, la décomposition hiérarchique est particulièrement attractive grâce à l'abstraction des informations extraites, c'est à dire la possibilité de cacher à un niveau les parties inutiles. Ce qui permet de raisonner sur chacun indépendamment des autres.*

2. *L'approche suppose un environnement quasi-statique entre les dates d'acquisition et d'action. Cette hypothèse n'est pas valable dans un environnement réel. Le contrôle peut s'avérer inefficace à cause de la longue durée séparant les instants d'acquisition et d'action. Le robot ne peut donc pas prendre en compte en temps réel son environnement. De nouvelles approches ont été introduites pour améliorer la réactivité*[1] *: aux niveaux bas pour plus de réactivité dans l'action, et au niveau haut pour*

1. La réactivité signifie que le système détecte et donne les réponses appropriées pour se prendre en charge.

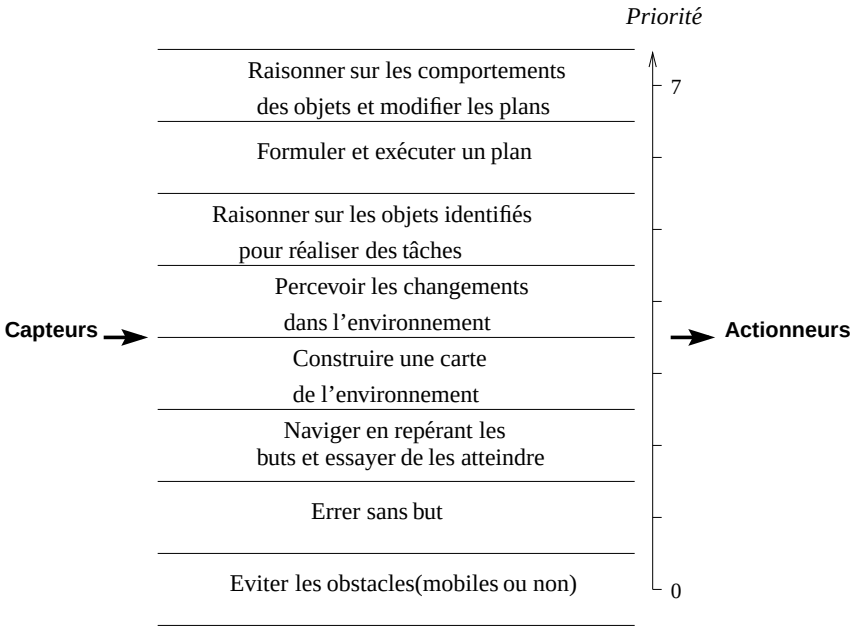


FIG. 1.2-5 – *L'architecture réactive par "subsumption" de Brooks.*

un raisonnement réactif.

1.2.2 Les architectures purement réactives

Les architectures purement réactives abandonnent le concept centré sur la modélisation de l'environnement et le raisonnement, au profit d'une approche distribuée privilégiant un lien étroit entre perception et action.

La réactivité à base de comportements

Dans cette approche l'activité du robot est modélisée par un ensemble de comportements hiérarchiques (figure 1.2-5) plutôt qu'une hiérarchie fondée sur une abstraction des données. L'action produite est directement liée à la détection d'un événement et le robot ne fait que réagir, en permanence, aux modifications de son environnement, sans planifier ni tenir compte des conséquences de ses actions. Le contrôle n'est alors que le fruit d'interactions entre les différents comportements actifs.

Brooks [Bro89] a proposé un système composé d'un ensemble de modules communicants qui s'exécutent d'une manière asynchrone et qui sont rangés en couches suivant une hiérarchie de comportements. Chaque module a accès aux capteurs et aux actionneurs, un comportement complexe est obtenu par combinaison des comportements élémentaires. Ceux situés en haut de la hiérarchie ont la capacité d'inhiber les entrées ou les sorties d'un comportement inférieur.

Cela ne veut pas dire réflexe, qui signifie plutôt la réaction directe et rapide entre les capteurs et les actionneurs sans réflexion.

Même avec un schéma très simple de priorités, il est souvent difficile de déterminer l'échelle des priorités des comportements concurrents. Ce qui implique des décisions arbitraires lorsque plusieurs comportements incohérents arrivent et ceci à cause de l'inexistence de données centralisées pour représenter l'état de l'environnement. De plus, une nouvelle conception est souvent nécessaire pour changer de comportement. Enfin, la spécification d'un objectif et la vérification restent très difficiles.

Des langages ont été développés pour faciliter la conception des architectures comportementales, par exemple le langage REX [KR89], le langage ALFA [Gat91] et le langage de comportement de Brooks [Bro89].

La réactivité à base de comportements mélangés

Elle constitue une nouvelle direction basée sur la fusion d'un ensemble de règles de logique et de règles floues. Des comportements complexes sont obtenus en mélangeant des comportements simples [SKR93]. Le mélange est constitué de fonctions désirables et de règles de contextes. Le fonctionnement du système est directement décrit à travers l'utilisation de règles linguistes écrites en langage naturel, ce qui facilite la conception et l'interaction avec le niveau symbolique. Cependant, le problème majeur réside dans le fait que l'on ne puisse pas vérifier formellement les comportements spécifiés [Gar95].

Les schémas moteurs d'Arkin

Arkin [Ark87] s'appuie sur la notion de schémas issue de travaux menés en psychologie et en neurologie pour spécifier des comportements génériques. Chaque comportement (schéma) est une entité générique qui décrit une interaction entre la perception et une action particulière. Un schéma moteur propose un déplacement sous la forme d'un vecteur représentant la direction et la vitesse du prochain déplacement devant être effectué par le robot. Les sorties de tous les schémas sont ensuite combinées par sommation vectorielle pour déterminer le déplacement effectif du robot.

1.3 Les architectures hybrides

Bien que hiérarchiques, les architectures hybrides se distinguent de l'approche purement hiérarchique dans la manière dont le sous-système réactif est guidé par le sous-système délibératif. Le sous-système réactif, qui est exécuté continuellement, parallèlement, et indépendamment du sous-système délibératif, peut répondre rapidement par réflexe aux événements externes (comme par exemple à l'approche des obstacles).

Une classe particulière d'architecture hybride est celle à trois niveaux. Ces niveaux sont respectivement de haut en bas : le *niveau délibératif*, le *niveau séquenceur* et le *niveau réactif*. Parfois un quatrième niveau d'abstraction est ajouté pour les asservissements [Pay86].

Pour comparer quelques systèmes, nous les avons regroupés suivant leur place dans les trois niveaux d'une architecture hybride.

1.3.1 Le niveau réactif

Le niveau réactif permet de faire la transition entre le raisonnement symbolique et le contrôle. Il peut aussi combiner les comportements séparés. C'est cette couche qui accède aux capteurs et aux actionneurs du robot et reste très peu en contact avec l'environnement. Un schéma de commande-contrôle complexe nécessite à ce niveau des réactions très rapides et réflexes pour l'évitement d'obstacle, la recherche de cible, la prise d'objet, etc.

1.3.2 Le niveau séquenceur

Le niveau séquenceur nécessite une planification réactive de tâches. Il sélectionne les tactiques appropriées en utilisant des règles en fonction du contexte. A partir d'un ensemble d'actions pré-écrites (comportements, opérateurs), le niveau séquenceur sélectionne la plus appropriée et l'exécute en accord avec les actions précédentes. L'exécution, qui peut nécessiter des appels récursifs, conduit à l'activation ou à la terminaison des fonctionnalités données par la couche réactive. Ceci nécessite la surveillance des conditions de terminaison et des échecs d'exécution.

Ce niveau est intermédiaire dans les architectures hybrides à trois niveaux. Son rôle est d'initialiser, de terminer et de surveiller les activités du niveau en dessous. Afin de terminer ces activités, le niveau séquenceur est consulté pour surveiller et reconnaître le succès ou l'échec des activités.

A ce niveau, deux techniques sont utilisées :

La planification réactive Elle est basée sur l'utilisation des plans temporels, pour la décomposition des tâches en sous-tâches, et des heuristiques de sélection. Les plans temporels peuvent être très complexes. Un plan est proposé au démarrage et plus de précision sur la spécificité de ce plan est donnée quand le système est sur le point de l'utiliser. Ceci permet de retarder les décisions et de faire plusieurs hypothèses sur l'environnement au moment de la construction du plan. Procedural Reasoning System (PRS) [GL87] et Reactive Action Package (RAP) [Fir89] sont deux représentations de la planification réactive que nous allons détailler par la suite.

Les plans compilés Au lieu d'utiliser un planificateur temps réel qui propose un plan à partir de la situation courante pour atteindre le but, la technique des plans compilés considère toutes les situations possibles (ou la plupart d'entre elles) et propose les actions appropriées. Un exemple de cette approche est les plans universels [Sch87]. Ce sont des arbres de décisions qui représentent l'état courant de l'environnement et l'action à suivre. Ils sont créés hors ligne en prenant le but comme condition de transition. Des opérateurs sont utilisés pour générer les plans. Bien que les plans compilés fournissent rapidement l'action optimale, leur action dans le monde réel est un problème dû à l'augmentation exponentielle des actions dans les arbres de décisions. Néanmoins, l'idée peut être bien exploitée pour former des plans qui couvrent presque tous les états de l'environnement (Plans Presque Universels) en laissant le reste des situations aux méthodes de *délibération* [Sch89].

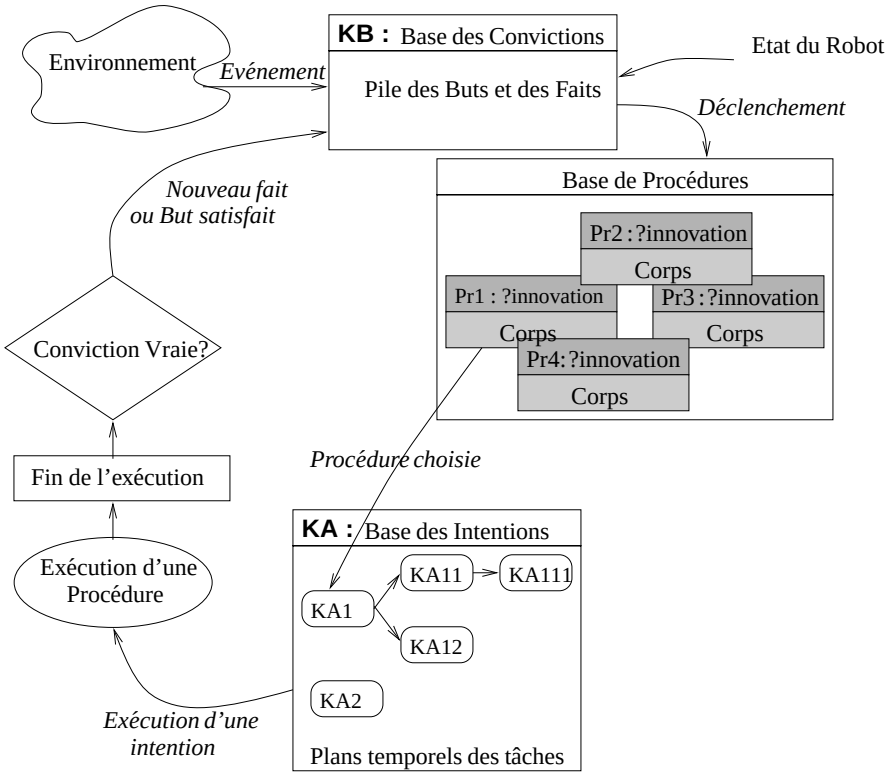


FIG. 1.3-6 – *Mécanisme des* PRS.

Le système PRS a été choisi pour montrer une technique de fonctionnement d'un séquenceur. D'autres systèmes RAP, TCA et PEM sont aussi brièvement décrits et comparés.

P**rocedural** R**easoning** S**ystem (**PRS**)**

A partir d'un plan partiellement défini au départ, le système PRS [GL87] élabore un plan plus spécifique et précis quand le robot est sur le point de l'exécuter. Ce qui permet de donner au robot un comportement pas strictement pré-défini au départ et d'éviter de faire plusieurs hypothèses sur l'environnement. Le robot évolue vers l'objectif, suivant ses *"intentions"* et ses *"convictions"*. Il essaie tout d'abord de satisfaire les intentions, représentées par la chaîne d'actions qui suit celle en cours (plans temporels). En même temps les intentions peuvent être modifiées par une conviction qui correspond au modèle de l'environnement. Ainsi, le robot aura des actions réactives, puisqu'il sera capable d'interrompre immédiatement n'importe quelle action et de proposer un autre plan d'actions.

A un instant donné, le système PRS fournit des intentions, sous forme d'objectives actuels, qui dictent le comportement futur du robot. A chaque objectif, un contexte particulier est associé. Ce contexte est directement lié à l'état courant de l'environnement et à l'historique des tâches que le robot avait déjà exécutées. D'un autre côté, chaque intention du robot est seulement valide dans une configuration environnement/robot donnée. Les changements imprévus dans l'environnement peuvent créer de nouvelles convictions et rendre l'intention courante invalide.

Mécanisme de fonctionnement : Le système PRS (figure 1.3-6) utilise deux bases de connaissances :

K**nowledge** B**ase (**KB**) :** Cette base contient l'ensemble des "convictions" du robot à cet instant précis. Pour cela, elle utilise le modèle de l'environnement et les données de l'état interne du robot.

K**nowledge** A**reas (**KA**) :** Cette base contient les "intentions", ou les plans temporels, pour atteindre certaines tâches ou buts. Une KA est une procédure composée d'un **corps** et des **conditions d'innovations**.

Grâce aux conditions d'innovations, le système vérifie que le robot est conduit par ses convictions en testant si un fait est présent dans KB (i.e. que ce fait est une conviction) avant d'exécuter la KA. Il s'assure aussi que le comportement du robot est conduit par les intentions, en vérifiant qu'à la fin de l'exécution d'une KA, que la conviction est toujours vraie, c'est à dire présente dans KB.

Le PRS commence par sélectionner une KA, donc une intention, et essaie de les satisfaire. En même temps, le système doit respecter les convictions dans la KB et rester sensible aux changements du monde extérieur et peut interrompre une action en cours suivant les événements. Ceci peut se faire soit en faisant appelle à une autre KA ou bien en faisant un changement direct dans la pile des buts. La seconde solution permet au système de changer complètement de cible et d'objectif quand une situation particulière le demande.

Remarques 1.3.1

1. *Les* PRS *ont une sémantique très puissante qui force le concepteur à entrer dans les détails de la dynamique des actions.*

2. *L'exécution parallèle des actions est possible mais une implémentation sur une architecture distribuée n'est pas supportée.*

3. *Plusieurs outils* SRI-PRS, C-PRS *[ACS],* UM-PRS *en* LISP *et* C *sont associés au système pour faciliter l'implémentation.*

4. *Le système a été utilisé dans un contrôle d'une navette spaciale de la* NASA *et en tant que séquenceur dans les architectures hybrides* FLAKEY *[SKR93] et* LAAS *[ICAR95].*

R**eactive** A**ction** P**ackages (**RAP**)**

Le RAP est un processus d'autonomie qui suit son but jusqu'à ce qu'il l'atteigne ou jusqu'à ce que toutes les possibilités pour l'atteindre soient épuisées [Fir87]. Dans le système RAP, la décomposition en sous-tâches, la supervision et la philosophie de traitement des exceptions sont organisées en unités discrètes. Une RAP définit des méthodes de décomposition des tâches en sous-tâches (i.e détermination de l'enchaînement des tâches élémentaires pour faire la tâche) et des méthodes de traitement des exceptions pour répondre aux changements dans l'environnement. La méthode est entièrement basée sur l'état courant de l'environnement sans anticipation et sans prendre d'hypothèses sur l'environnement.

Une RAP peut se référer hiérarchiquement à d'autres RAPs. Son mécanisme d'exécution est décidé comme suit : à partir d'une file d'attente représentant l'ordre d'exécution des RAPs, l'ordonnancement est établi suivant le modèle de l'environnement et la situation actuelle. Une RAP est soit exécutée, si elle se trouve en tête de la file d'exécution, soit recalée dans la file (sub-RAPs).

Remarques 1.3.2

1. *Le mécanisme d'exécution des* RAP*s est très bien défini par la situation actuelle indépendamment du futur alors que les dynamiques des* PRS *sont essentiellement déterminées par les* KA*s.*

2. *L'inconvénient des* RAP*s est leur incapacité à exécuter des tâches parallèles et à être implémenter sur une architecture distribuée [Fir95].*

3. *Les* RAP*s ont été utilisées dans la commande d'un Aspirateur autonome et comme séquenceur dans l'architecture hybride* ATLANTIS *.*

T**ask** C**ontrol** A**rchitecture (**TCA**)**

En haut de la hiérarchie du TCA [Sim90], on trouve la représentation de la relation entre tâche et sous-tâches. Cette représentation, appelée *arbre de tri*, a des buts, des commandes exécutables et des messages de surveillance. Le TCA construit l'arbre automatiquement : chaque fois qu'un but, une commande, ou un message de surveillance est émis, le système crée un noeud et lui ajoute des sous-noeuds. Des contraintes temporelles entre les noeuds sont utilisées pour enchaîner les tâches de planification et d'exécution : les messages sont empilés jusqu'à ce que les contraintes temporelles soient satisfaites. Cette combinaison hiérarchique de tâches et de contraintes temporelles forment les plans de représentation du système TCA.

Remarques 1.3.3

1. *Le* RAP *a un comportement plus réactif et plus temps réel que le* TCA*. Par contre le* TCA *s'occupe plus des interactions avec l'environnement pour le traitement des tâches complexes. En général, sur le spectre réactivité/délibération, les systèmes utilisant le* TCA *ont un comportement plus délibéré que ceux qui utilisent le* RAP *;*

2. *Le* TCA *supporte une architecture distribuée et permet une exécution parallèle de tâches ;*

3. *Le* TCA *a été intégré sur différentes plates-formes expérimentales [Sim90, Sim95] :*

   – *contrôle de la marche d'un robot a six pattes (Ambler) pour l'exploration des planètes ;*

   – *contrôle d'un robot (Ratler) prototype d'exploration lunaire ;*

   – *contrôle des robots mobiles d'intérieur (Hero) qui ramasse des tasses au moyen d'un bras manipulateur et Xavier qui explore l'environnement.*

Planning, Execution and Monitoring (PEM)

Il est basé sur la modélisation de contrôle de flot d'informations [HR92]. Les entités PEM sont des triplets qui ont les fonctionnalités de planification, d'exécution et de surveillance des actions (comportement, tâche). Toutes ces fonctionnalités sont surveillées par un Meta-contrôleur. Les PEM sont organisées en hiérarchie et leur exécution peut entraîner une décomposition d'une tâche en sous-tâches. Le principe dans le modèle PEM est le contrôle de processus, i.e les dynamiques de la planification, l'exécution, et la surveillance.

Remarques 1.3.4

1. *L'architecture* PEM *n'est encore supportée par aucun outil, ne permet pas l'exécution parallèle de tâches et n'autorise pas une implémentation distribuée.*

2. *Le système n'a été validé que d'une façon très primaire dans le contrôle d'un manipulateur autonome [RTJV94].*

1.3.3 Le niveau délibératif

C'est l'unique niveau qui peut assurer la modélisation symbolique de l'environnement. Plusieurs architectures délibérées ont été proposées. Les travaux d'Albus [Alb91] et de Meystel [Mey88] utilisent l'idée de la décomposition "Top-Down" des tâches en sous-tâches, pour la spécification des activités, courantes et futures, et des contraintes (temporelle, ressource). Les activités dans cette couche correspondent à une stratégie de planification à long terme et à d'éventuelles adaptations de plan au cours du temps (planification temporelle). Pour cela des techniques classiques de l'intelligence artificielle et de raisonnement pour la planification, l'enchaînement et l'optimisation des ressources sont utilisées. Les planificateurs utilisés à ce niveau sont IXTET [GL94a] et SPIE, SPIE2 [WMLW94, WM95].

IndeXed TimE Table (IXTET)

C'est un planificateur temporel à ordre partiel [GL94a, GAC87].

Pour engendrer les plans, IXTET manipule des PRS. Les plans sont produits sous la forme de treillis où les tâches, affectées en fonction du contexte, sont partiellement ordonnées suivant :

– des contraintes temporelles numériques telles la durée minimale ou maximale, ou des événements extérieurs datés (par exemple la tâche T_1 doit être accomplie à l'heure H) ;

– des contraintes temporelles symboliques (e.g la tâche T_1 doit être exécutée [avant, après, ...] la tâche T_2).

TAB. 1.3-1 – *Quelques architectures hybrides à trois niveaux.*

Système	Quelques applications
3T [BK94] - `planificateur` : AP - `séquenceur` : RAP	- contrôle de sous-marins, - navigation de robots mobiles (suivie de trajectoire et évitement d'obstacles) - Les manipulateurs spaciales ARMSS et EVA
ATLANTIS [Fir95] - `plannificateur` : simulateur - `séquenceur` : RAP avec des sémaphores - `réactif` : ALFA	- navigation externe de robots mobiles
FLAKEY [SKR93] - `séquenceur` : PRS - `réactive` : comportements mélangés	- missions de navigations complexes de robots mobiles
architecture du LAAS [CADL92, ICAR95] - `Planificateur` : IXTET - `séquenceur` : PRS	- navigation externe du robot mobile ADAM pour l'exploration des planettes, - application multi-robots
architecture de PAYTON [Pay86] quatre niveaux - `réactif` : deux niveaux	- navigation dans un environnement simulé
SSS [Con92] - `Planificateur` : pas un vrai - `séquenceur` : symbolique - `réactif` : - un pour les asservissements - un autre comportemental	- navigation interne de robots mobiles,

Le second composant de ce système est l'interpréteur de plans, dont le rôle consiste à gérer l'exécution du treillis produit par le planificateur. A chaque étape on doit vérifier le respect des contraintes spécifiées et maintenir à jour une description de l'état du monde. Cette description permet de vérifier la cohérence de la situation réelle avec la situation planifiée (les actions produites sont bien les actions désirées).

1.4 Les services d'un environnement de programmation robotique

La conception du contrôle et la programmation d'un robot nécessite la prise en compte de plusieurs domaines comme la théorie de l'automatique, la mécanique, la théorie des systèmes temps réel, les systèmes d'exploitation, la programmation des cartes d'entrée/sortie ("I/O Drivers") et d'autres encore qui dépendent de la nature de l'application robotique.

Ceci rend le développement des programmes d'applications et leurs maintenances chers. Ce coût peut être réduit en fournissant à l'utilisateur un environnement de programmation adéquat.

Nous pouvons définir un environnement de programmation robotique comme l'ensemble des ressources logicielles nécessaires à l'exécution d'une application robotique. Les systèmes d'exploitations temps réel classiques [NSH89, SSK89, Win93, Sof89, Rea93] fournissent des fonctions de base (création, suspension, communications entre les tâches, ...) mais ne permettent pas de construire d'une manière structurée une application de contrôle-commande. Le contrôle de flot, les réponses aux événements, l'échantillonnage des données, la communication à travers le réseau entre les composantes du système, les interfaces d'utilisation, ... sont tous à la charge du programmeur. La ressource logicielle de base dans un environnement de programmation est le(s) langage(s) de programmation utilisé(s). Ce langage doit :

- permettre une **programmation hiérarchisée** en disposant de plusieurs couches accessibles par des utilisateurs de compétences différentes ;
- permettre une **portabilité** future avec le minimum de modifications du code ;
- permettre de **garantir** que l'exécution du code produit est conforme à celui spécifié. Généralement, la théorie des automates est utilisée pour modéliser l'exécution. L'une des propriétés importantes des automates est leur **déterminisme** ;
- produire un code temps-réel capable de **réagir** à des événements **temporisés** externes (gestion des interruptions). En effet, le temps apparaît dans ce cas comme une contrainte logique et non comme un critère de performance ;
- permettre d'exprimer le **parallélisme** ;
- disposer d'un confort à la spécification non payable à l'exécution. Pour cela il faut **optimiser** les ressources ;

  L'environnement doit également pouvoir :

- combiner plusieurs programmes **distribués** sur une architecture physiquement répartie ;
- simuler les lois de commande et diagnostiquer les variables à commander pour mieux les ajuster.

1.5 Quelques environnements de programmation

1.5.1 CONTROLSHELL

CONTROLSHELL [SCPC95] est un environnement qui combine une construction basée sur le flot de données, une programmation d'état basée sur les événements, un exécutif

temps réel et une connection transparente par réseau. Les motivations de l'existence du système sont la *réutilisation* du code et son *partage* par plusieurs processeurs. Par construction d'unités fonctionnelles, les concepteurs peuvent prendre les blocs déjà conçus pour les réutiliser sur plusieurs autres applications. L'idéal est de construire les applications sans écrire de nouveau code en liant seulement les librairies existantes.

CONTROLSHELL privilégie la pratique, son but est d'implémenter des applications et non de vérifier formellement des contraintes temps-réel, il ne peut garantir que la machine d'états finis n'a pas de "Deadlock". Par contre, CONTROLSHELL assure le fonctionnement des systèmes cibles par des simulations et des validations du code temps réel sur un matériel réel. Plusieurs outils sont fournis pour cela. L'OS de base est VXWORKS, le système de communication temps réel est le NDDS, le système de visualisation temps réel et de collection des données pour l'analyse des performances est le STETHOSCOPE. La conception est faite dans "Aerospace Robotics Laboratory" à Stanford University et conjointement avec "Real-Time Innovation, Inc.".

Les éléments de base sont des *modules* de programme réutilisables. De nouveaux modules peuvent être ajoutés à une librairie qui contient des éléments prédéfinis (filtres, générateurs de trajectoires, ...). A partir de ces modules et à l'aide d'un éditeur graphique, des *composantes* sont construites en spécifiant les fonctions d'initialisation et de terminaison ainsi que les données d'entrées et de sorties. Une composante représente donc un schéma de flot de données. Les stratégies d'activation/désactivation, flot de contrôle, sont programmées par un autre éditeur graphique sous forme d'un graphe d'états.

CONTROLSHELL a été utilisé dans plusieurs applications relatives aux robots manipulateurs. Le système est très facilement reconfigurable et peut prendre en compte les événements asynchrones durant l'exécution périodique des tâches. Cependant ces tâches sont relativement simples.

1.5.2 CHIMERA 3.0

La programmation orientée objet [Boo86] est une technique permettant la réutilisation du code sur plusieurs applications différentes. On la retrouve dans plusieurs environnements RIPE [ML91] pour la planification et la programmation des applications robotiques et dans CHAOS [SPB87] pour le support des objets temps réel. L'environnement le plus puissant est certainement celui développé à l'Université de Canergie Mellon (CMU) et supporté par l'U.S. Army [SVK93]. Ce système, qui est très proche du système STER [AM91], combine :

- le concept orienté objet pour fournir des méthodes cohérentes de réutilisation de code sur plusieurs applications et un code totalement indépendant de l'architecture matérielle ;
- un automate pour le contrôle de la partie discrète du système.

La version 3.0 de CHIMERA [SKHK89] est basée sur un ensemble de composantes définis comme des objets appelés : *"port-based objects"*. Un *"port-based object"* inclut un état interne, son code, les données, les ports d'entrée/sortie et les ports de ressources.

Les ports d'entrée/sortie permettent de connecter plusieurs objets alors que les ports de ressources sont utilisés pour l'interaction avec un opérateur.

Une tâche temps réel est associé à chaque objet. Elle est gérée par des primitives semblables à celles qu'on trouve dans un système d'exploitation temps réel commercial tel que pSOS [Sof89] avec la possibilité de choisir entre diverses politiques d'ordonnancement et d'être réalisée en utilisant des priorités fixes ou dynamiques. Les objets ont quatre états possibles : NOT-CREATED, ON, OFF et ERROR. Un *"job"* est obtenu en connectant plusieurs objects.

L'utilisateur dispose d'un environnement, ONIKA [Ger93], constitué autour d'une (ou plusieurs) station de travail pour la spécification et le développement de programmes temps réel ainsi que la simulation des lois de commande. Les programmes sont censés être exécutés sur un système informatique temps réel constitué de deux couches. La première s'occupe de la supervision et la deuxième des asservissements.

Le système a été expérimenté sur plusieurs robots manipulateurs [KKT84] et sur un robot mobile autonome pour les manipulations spatiales [BFK90]. Les applications ont été programmées sous le même environnement et certains modules de programmes ont été réutilisés sur les divers robots.

1.5.3 Autres environnements

Des systèmes très semblables à CHIMERA 3.0 ont été développés.

IBM a conçu l'environnement SPARTA [ISK89] dans ses centres de recherche à Zurich et à Yorktown. Un langage de programmation haut niveau CS (Control System language) [IS85] a été développé pour la décomposition d'une tâche en plusieurs sous-tâches. L'environnement permet la génération automatique du code temps réel, le "debbugage", le chargement dynamique d'algorithmes pendant l'exécution et la vision des données en temps réel.

De même, au MIT (Massachusetts Institute of Technology) l'environnement CONDOR [NSH89] a été testé sur une architecture de contrôle très puissante composée d'une main articulée avec 16 liaisons et 2 actionneurs par liaison.

1.6 Conclusion

Nous avons présenté dans ce chapitre les différentes approches utilisées dans les contrôleurs autonomes. La réactivité est la méthodologie de base. D'une part, la réactivité au niveau haut peut traiter des tâches complexes en les décomposant en plusieurs sous problèmes traitables. La planification peut aider le robot à éviter les pièges locaux dans l'environnement, et peut aussi diminuer les risques tout en augmentant la robustesse. Par contre, la décomposition en tâches n'est bonne que si le robot a les modèles et les informations nécessaires. Souvent, la décision est ajournée jusqu'à ce que le robot accède à l'information voulue. D'autre part, la réactivité au niveau bas donne au robot une sécurité en désactivant le système suivant la situation dans l'environnement. Mais il est plus difficile pour ces architectures de supporter les comportements interactifs, sauf s'ils sont connus et spécifiés à

l'avance. De plus, l'absence de mécanismes de raisonnement sur les actions et leurs conséquences limitent la complexité des buts qui peuvent être pris en compte : il nous semble difficile d'implanter un comportement générique et sophistiqué tel "Va_vers_le_parking(X)". Ainsi, l'ajout de structures aux comportements réactifs nous apparaît indispensable dans l'architecture de contrôle. C'est dans cette optique que les architectures hybrides ont été conçues combinant raisonnement et mécanismes de réactivité dans les actions.

Par la suite, nous utiliserons le contrôleur hybride ORCCAD, qui en association avec des ressources logicielles de spécification et de vérification permet une programmation modulaire, évolutif et sécuritaire des applications robotiques complexes. Nous compléterons le système ORCCAD d'un ensemble de fonctionnalités spécifiques pour les robots mobiles en général et pour les véhicules type voiture en particulier.

Chapitre 2

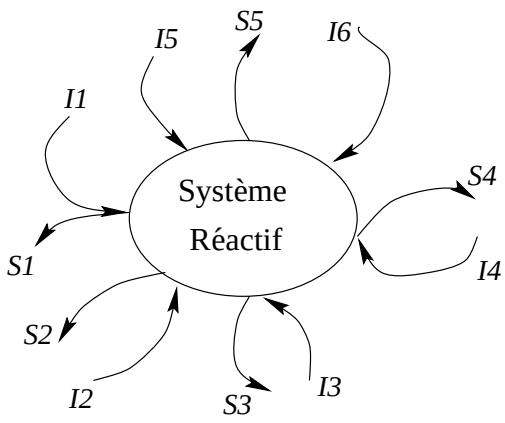
Outils de Spécification et de Vérification des Systèmes Réactifs

DANS le premier chapitre, nous avons vu que la *réactivité* est une caractéristique fondamentale des systèmes robotiques. Un ou plusieurs sous-systèmes réactifs sont associés pour prendre en charge la navigation d'un robot. Cependant, un système temps-réel aussi complexe qu'un robot autonome n'est pas constitué que par des parties réactives. Un noyau interactif pour la gestion des interruptions, des communications, de l'interface avec l'environnement est associé à ces noyaux réactifs pour exécuter les actions futures. Ceci fait que la programmation de ces systèmes pose de nombreux problèmes tels que la satisfaction de contraintes temporelles (durée et temps de réponse), la sûreté de fonctionnement... Les erreurs de conception constituent l'une des sources de défaillance les plus probables. Nous pensons qu'il est nécessaire de recourir à des méthodes formelles de vérification dans le processus de spécification-validation. De plus, si la complexité des algorithmes nécessite une parallélisation sur une architecture matérielle distribuée pour satisfaire la contrainte temps réel d'autres problèmes comme l'ordonnancement et la répartition des programmes sur les différents processeurs, la synchronisation, la communication entre les processeurs...deviennent des problèmes plus durs à résoudre.

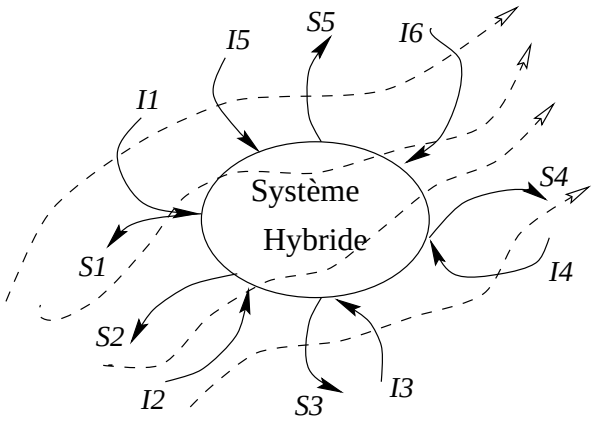
Nous préciserons dans ce chapitre les méthodes générales de spécification et les techniques formelles de vérification des systèmes réactifs.

Le plan suivi dans ce chapitre est le suivant :

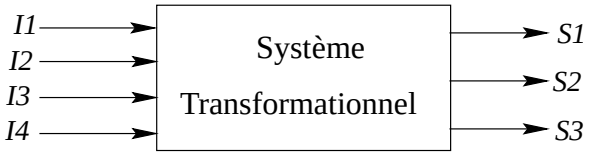
- ☞ la section 2.1 présente la nature et les caractéristiques des systèmes réactifs;
- ☞ nous présenterons dans la section 2.2 les méthodes classiques de spécification. Une analyse sur les avantages et les inconvénients des uns par rapport aux autres sera donnée;
- ☞ dans la section 2.3, nous présentons l'approche synchrone et l'hypothèse de son fonctionnement;



(a) Système réactif.



(b) Système hybride.



(c) Système transformationnel.

FIG. 2.1-1 – *Classification des systèmes informatiques.*

- ☞ après nous présenterons, dans la section 2.4, les deux familles de langages synchrones qui traitent les problèmes de la programmation temps réel d'une manière différente;
- ☞ nous parlerons dans la section 2.5 des modèles d'automates pour la vérification des systèmes réactifs. Des exemples de modèles seront donnés;
- ☞ dans les sections 2.6, les méthodes de vérification de ces modèles et les outils associés seront décrits d'une manière générale;

2.1 Nature des systèmes réactifs

D'après [HP85], un système *réactif* (figure 2.1-1(a)) fait partie d'une classe particulière de systèmes informatiques qui sont continûment en interaction avec leur environnement, sans maîtriser le rythme de ces interactions, en réagissant à des entrées par l'émission de sorties appropriées. Ces sorties dépendent non seulement des entrées à l'instant courant, mais aussi potentiellement de toutes les interactions qui précèdent cet instant. La plupart des systèmes de contrôle-commande sont des systèmes réactifs. L'absence de maîtrise, par le système, du rythme de ses interactions avec l'environnement est une caractéristique fondamentale des systèmes réactifs.

La seconde classe regroupe les systèmes *interactifs* qui interagissent aussi continûment avec leur environnement mais cette fois à une vitesse qui leur est propre. Par rapport aux

systèmes réactifs, ces systèmes ont la possibilité de mémoriser certaines entrées avant de les prendre en compte et cela est sans conséquence sur le déroulement du système. Un système d'exploitation est un exemple de système interactif.

La troisième classe représente les systèmes les plus classiques en informatique, ce sont les systèmes *transformationnels* (figure 2.1-1(c)) qui n'ont aucune interaction avec l'environnement. Ils prennent un ensemble de données au début, opèrent sur ces données des transformations, et produisent à la fin de l'exécution un ensemble de résultats. Un compilateur en est un exemple.

Plus récemment, les systèmes *hybrides* (figure 2.1-1(b)) ont été introduits [MMP92], pour modéliser l'évolution d'un système réactif, à caractère foncièrement *discret*, dans un environnement physique qui, lui, aura plutôt un caractère *continu*. Par exemple, l'environnement pourrait être caractérisé par des variables continues comme la température, et le système réactif (plongé dans cet environnement), serait un programme dont le rôle est de *contrôler* les variations de la température.

Caractéristiques des systèmes réactifs Les systèmes réactifs sont des systèmes :

① *déterministes* car leurs sorties sont entièrement déterminées par les valeurs et les instants d'occurrence de leurs entrées. Il s'ensuit que le comportement d'un système réactif doit être reproductible;

② *parallèles* car il est naturel de concevoir un système réactif comme un ensemble de composants s'exécutant en parallèle et coopérant à la réalisation du comportement global souhaité.

De plus, ils sont soumis à des contraintes :

➤ *de fiabilité* vu que ces systèmes interviennent dans le contrôle de dispositifs critiques. Il est donc important de pouvoir vérifier formellement certaines propriétés du système;

➤ *temporelles.* Un système réactif temps-réel possède en plus la notion de temps. Cela se traduit par l'ajout de contraintes temporelles directes ("la boucle d'asservissement doit être exécutée au moins toutes les secondes") ou indirectes ("devant un obstacle, le robot doit s'arrêter") que le système doit respecter.

2.2 Approches classiques de programmation

Les premiers systèmes réactifs ont été réalisés à l'aide de machines analogiques et des circuits à relais. Les progrès considérables dans le domaine des microprocesseurs ont fait évoluer la mise en oeuvre de ces systèmes vers la programmation en langage machine. Ce type de programmation reste efficace car ses instructions sont directement liées à l'architecture des microprocesseurs. Cependant, vu la complexité des applications actuelles, la situation a rapidement évolué vers les techniques de programmation haut niveau. Une étude comparative des différentes approches est donnée dans [BB91].

TAB. 2.2-1 – *Les approches de programmation.*

Approche	Efficacité	Déterminisme	Extension	Preuve	Concurrence
Automates déterministes	+	+	-	+	-
Réseaux de Pétri	=	-	-	=	+
Tâches séquentielles	=	-	-	-	-
Processus communicants	=	-	=	-	+
Approche synchrone	+	+	=	+	+

Le signe + signifie que l'approche est remarquable pour la propriété considérée.
Le signe = signifie que l'approche est moyenne pour la propriété considérée.
Le signe - signifie que l'approche est mauvaise pour la propriété considérée.

2.2.1 Tâches gérées par un système d'exploitation temps réel

C'est la méthode la plus utilisée. Une application est formée par un ensemble de programmes séquentiels écrits avec un langage classique comme le langage C ou le PASCAL. Des primitives d'un système d'exploitation temps réel sont utilisées pour gérer la communication et la synchronisation entre ces programmes. On trouve dans le commerce plusieurs systèmes d'exploitation temps réel, tels que VXWORKS, PSOS, ...pour diverses architectures de processeurs. Ces systèmes sont généralement non-déterministes à cause de l'ordonnancement arbitraire des tâches créées, sauf dans le cas où l'application est réduite à une séquence triviale. Ce non-déterminisme rend le "debugging" très difficile étant donné qu'il est difficile de "rejouer" le programme pour une même séquence d'entrées.

2.2.2 Automates d'états finis, réseaux de Petri et Grafcet

Les automates d'états finis ont de nombreux avantages : ils sont déterministes, efficaces sur une architecture non distribuée, et ils peuvent être automatiquement analysés par des outils de vérification. Cependant, ils sont très peu structurés et manquent de primitives de haut niveau pour exprimer le parallélisme par exemple. Un automate induit un ordre total sur les données manipulées, ce qui limite sa distribution sur une architecture parallèle. En plus, un léger changement dans la spécification peut entraîner une re-conception complète de l'automate. Leur programmation reste bien adaptée pour les petites applications de contrôle. A partir d'une certaine taille, la compréhension du fonctionnement de l'automate devient très difficile.

Par rapport aux automates, les réseaux de Petri et le Grafcet peuvent exprimer le parallélisme, mais ils manquent de modularité pour programmer des applications de grandes tailles. Ils sont, aussi, bien adaptés pour les petites applications.

2.2.3 Langages de programmation parallèle

Des langages de programmation comme ADA [Spr83], OCCAM [Occ84] sont très bien structurés et peuvent exprimer le parallélisme. En plus des primitives de calculs numériques

que l'on trouve dans les langages classiques, ils possèdent des primitives de gestion de processus (création, suspension de tâches,...) des systèmes d'exploitation temps réel. Ce qui fait que les applications écrites à l'aide de ces langages seront indépendantes des systèmes d'exploitation, et théoriquement elles peuvent être portées d'une configuration matérielle à une autre sans être modifiées.

Cependant, plusieurs inconvénients sont associés à l'utilisation de ces langages. En effet, ces langages sont essentiellement asynchrones et non-déterministes [Ber89]. Pour un même programme et une même histoire d'entrée, plusieurs comportements différents à l'exécution sont possibles. Par conséquent, il n'est pas possible de "reproduire" un comportement pour effectuer des analyses et des tests. Enfin, il n'est pas possible de réaliser des vérifications formelles automatiques sur les programmes produits.

2.2.4 Conclusion

Des langages haut niveau déterministes intégrant la notion de parallélisme de description doivent être utilisés pour spécifier et programmer les parties réactives d'une application. L'approche synchrone a été introduite pour essayer de résoudre les problèmes rencontrés avec les méthodes classiques de spécification.

2.3 L'approche synchrone

Les langages synchrones [BB91] constituent une famille de langages de haut niveau, dédiés à la programmation de systèmes *réactifs*. Comme on l'a déjà évoqué, la plupart des applications *temps réel* sont réactives parce qu'elles sont directement influencées par l'environnement dans lequel elles évoluent et ont en plus une *contrainte temporelle* à respecter. L'approche synchrone s'applique pour la programmation de ces systèmes si l'hypothèse de synchronisme est valide. Cette hypothèse s'appuie sur l'exécution en temps nul. (voir figure 2.3-2).

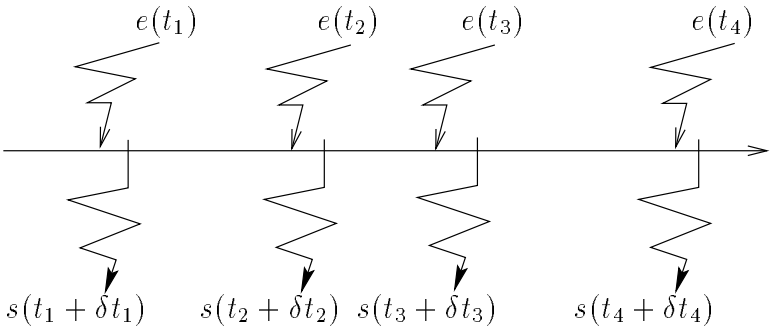


FIG. 2.3-2 – *L'hypothèse synchrone.*

Du point de vue interne, cette hypothèse est valable car la décomposition du système réactif en un ensemble de sous systèmes, qui s'exécutent en parallèle est calculée lors de la compilation et ses communications sont statiques.

Du point de vue externe, cette hypothèse est bien sûr idéale car le programme est confronté à un environnement de nature asynchrone où les entrées arrivent sans aucun ordre préétabli. Imaginons l'exécution comme une boucle infinie : "lecture des entrées", "calcul des sorties", "écriture des sorties", le synchronisme est valide si le temps de calcul des sorties δt_i est suffisamment rapide pour ne pas perdre les entrées. Il est par conséquent nécessaire, pour valider l'hypothèse de synchronisme, de savoir prédire le temps d'exécution de la phase "calcul des sorties" et de s'assurer qu'il est toujours plus petit que le délai minimal d'occurrence des entrées : c'est à dire que :

$$\forall i,\ \delta t_i < \min_i\{(t_{i+1} - t_i)\}$$

2.4 Les langages synchrones

Les langages synchrones partagent l'idée que les systèmes réactifs se décomposent en sous systèmes qui évoluent en parallèle et communiquent entre eux, afin d'obtenir le comportement global souhaité. On note que cette décomposition n'implique pas une répartition des sous systèmes sur une architecture distribuée. Ces composantes s'exécutent et communiquent entre elles en temps nul par *diffusion synchrone* [Ber89]. L'hypothèse de communication en temps nul est valable car toutes les communications sont statiques, prévues et calculées à la compilation.

L'hypothèse de synchronisme peut engendrer certains paradoxes, semblables aux problèmes de court circuit rencontrés dans les circuits électroniques : ces problèmes apparaissent dès que l'émission d'un signal dépend de sa réception dans le même instant, ou à des définitions cycliques, à des spécifications de comportements pour lesquels il n'existe pas de solution, à des comportements non déterministes, ou à des dépendances de données contradictoires. Les compilateurs des langages synchrones permettent de déceler ces problèmes dits : les *erreurs de causalité*. En même temps que la vérification "classique" sur la syntaxe, la détection des erreurs de causalité est effectuée en vue d'une vérification statique.

La compilation d'un programme écrit avec un de ces langages produit un automate booléen, temporisé ou hybride suivant le langage. Les vérifications formelles possibles du comportement de ce programme dépendent de la sémantique du langage. Une translation vers un langage cible (le langage C ou ADA) est également produit. Le code produit est ensuite compilé avec le compilateur standard sur une architecture du processeur cible. L'exécution de ce code a l'avantage d'être déterministe et d'avoir un comportement à l'exécution formellement vérifiable.

Les différents langages élaborés se distinguent essentiellement par les modèles des algorithmes qui facilitent l'expression et la résolution d'un problème particulier. Notons qu'il existe un format commun d'expression des automates pour tous les langages synchrones, ce qui permet de partager les outils de vérification indépendamment du langage utilisé. On peut en distinguer deux familles : les langages impératifs et les langages déclaratifs.

2.4.1 Les langages impératifs

Ces langages s'appliquent plus particulièrement aux systèmes dont les *aspects algorithmiques de contrôle sont prépondérants*. Le modèle flot de contrôle est un automate dont les sommets représentent les états et les arcs les transitions auxquelles sont associées des actions. Le contrôle des actions est défini par l'état et l'événement qui active la transition.

ESTEREL

C'est le plus ancien langage synchrone. C'est un langage à flot de contrôle de style impératif textuel qui décrit les actions (ou sorties) à effectuer sous l'impulsion d'événements. Il contient à la fois des structures de contrôle classiques et des structures spécifiques aux systèmes réactifs. Il est développé conjointement par l'Institut National de Recherche en Informatique et en Automatique (INRIA) et l'Ecole Nationale Supérieure des Mines de Paris (ENSMP). Il est commercialisé par les deux sociétés CISI-Ingénierie et Ilog et supporté par Simulog. Une description complète de ce langage peut être trouvée dans [BdS91] et à [Hal93].

Ses principales caractéristiques sont les suivantes :

- Comportement déterministe, même pour les branches parallèles,
- Communication par diffusion : tout le monde reçoit tous les messages en même temps, et de manière synchrone avec l'émission,
- Décisions instantanées : plusieurs émissions et réceptions pendant le même instant sont possibles,
- Il existe une forme d'asynchronisme réalisée à l'aide de la primitive `exec` (voir paragraph 2.4.3 [Par92] pour plus d'explication),

Le compilateur permet de faire :

① la vérification statique des erreurs de causalité;

② la production d'un automate à états finis codé avec le format `OC`[1]. Il est alors possible d'utiliser des outils de preuve;

③ la production d'un code séquentiel efficace;

1. Objet Code est un langage intermédiaire de déclaration sous forme d'automates séquentiels déterministes. Différents outils traitant ce code ont été développés. Cette étape est utilisée par la plupart des langages synchrone.

ARGOS **: Un langage graphique de composition d'automates**

ARGOS [Mar90] est un langage graphique basé sur le standard STATECHARTS [Har87]. Il utilise des automates parallèles et hiérarchisés et un jeu d'opérateurs sur les automates pour définir structurellement des automates complexes. Une description de ce langage peut être trouvée dans [Hal93]. Ses principales caractéristiques sont les suivantes :

- Emission de type diffusion des signaux de sortie des processus;
- Déterminisme d'évolution séquentielle : deux processus en parallèle évoluent de manière synchrone et déterministe;
- Non déterminisme de choix : si deux transitions à partir d'un même état du graphe sont franchissables en même temps, le choix de l'état suivant n'est pas déterministe;
- Programmation modulaire et hiérarchisée;

Le compilateur permet de :

① détecter les erreurs de causalité;

② créer un code `OC`

2.4.2 Les langages déclaratifs (à flot de données)

Dans ces langages, la donnée traitée est du type signal physique échantillonné (flots de données discontinues). Chaque flot représente une séquence (infinie) de valeurs d'un même type (entier, booléen, réel,...). Les instants de présence de ces valeurs sont donc discontinus et représentent l'horloge du signal. Deux notions sont indissociables : l'horloge et la suite de valeurs. Ces langages sont plus particulièrement adaptés aux problèmes où les *aspects algorithmiques liés aux flots de données sont prépondérants* comme par exemple en traitement de signal. Le graphe flot de données est composé de sommets qui représentent les actions et d'arcs qui représentent les transferts de données. Avec ce modèle on peut mettre plus facilement en évidence l'ordre partiel de transfert des données dans l'algorithme et trouver le parallélisme potentiel (en fonction des dépendances) pour distribuer l'algorithme sur une architecture multiprocesseurs.

LUSTRE

Le langage LUSTRE [HLR92] qui a été développé à l'Institut de Mathématiques Appliquées de Grenoble (IMAG) et commercialisé par la société Verilog, est actuellement utilisé dans l'industrie (avionique, nucléaire...). Ses principales caractéristiques sont les suivantes :

- Comportement déterministe;
- Optimisation et validation de propriétés sur le programme en utilisant la méthode des observateurs. Plus de détails sur cette méthode sont donnés dans le paragraphe 2.6.3;

- Programmation modulaire;

Le compilateur permet :

① le calcul d'horloges permet de vérifier la consistance globale du programme (problèmes de causalité, contraintes de présence horloge);

② la compilation en un automate à états finis : code OC;

③ la création d'un code séquentiel;

SIGNAL

Ce langage a été développé par l'Institut de Recherche en Informatique et Système Aléatoires (IRISA) et commercialisé par la société TNI. Une description complète de ce langage pourra être trouvée dans [GGBM91]. Ses caractéristiques sont :

- Comportement déterministe du code exécutable produit;
- Programmation hiérarchique;

Le compilateur permet de :

① Tester la causalité sur le graphe des dépendances de données et d'horloges;

② Créer un code efficace sur une architecture répartie ou non;

SYNDEX

SYNDEX est un environnement de programmation graphique interactif pour les applications de traitement du signal et d'automatique s'exécutant en temps réel sur des machines multi-processeur. Il offre les fonctionnalités suivantes :

- interface avec SIGNAL [BLG+94] pour la spécification et la vérification de l'algorithme d'application afin d'obtenir des programmes fiables;
- spécification et dimensionnement de la machine multi-processeur en vue d'optimiser le matériel;
- placement-ordonnancement optimisé de l'algorithme d'application sur multi-processeur avec évaluation et visualisation de ses performances;
- génération de l'exécutif distribué en temps-réel, déchargeant l'utilisateur au maximum des tâches lourdes de programmation bas niveau.

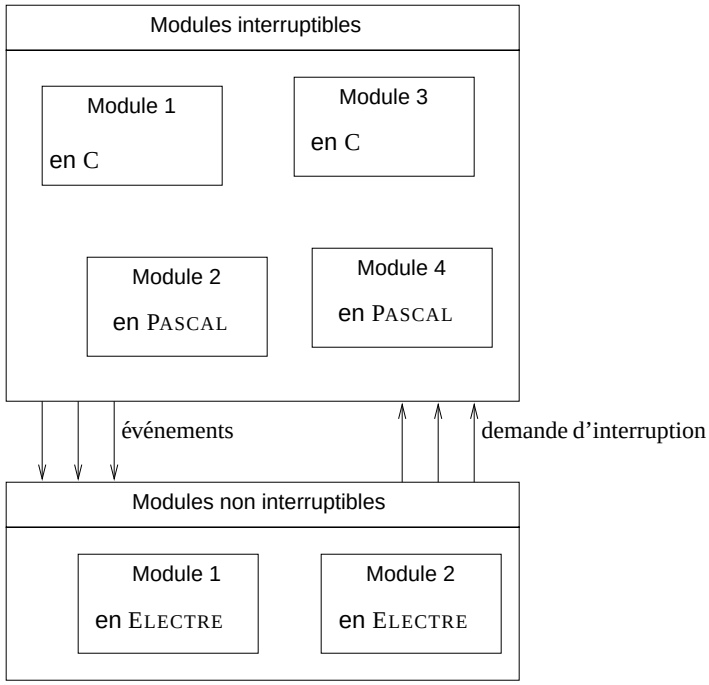


FIG. 2.4-3 – *Modèle d'un système utilisant* ELECTRE.

2.4.3 Aspects asynchrones

L'hypothèse synchrone n'est pas nécessairement valable dans tous les systèmes réactifs, en particulier lorsqu'ils font intervenir des tâches qui s'exécutent en un temps supérieur à celui séparant deux occurrences d'événements. Ce besoin a amené les concepteurs du langage synchrone ESTEREL à introduire la primitive *exec*. Sa fonction est de lancer des tâches externes et de ne laisser pour la partie réactive que la gestion de début et de fin d'exécution, ainsi l'hypothèse synchrone est conservée.

Le langage ELECTRE

Le langage réactif asynchrone ELECTRE [RCCE92] est développé au Laboratoire d'Automatique de Nantes (LAN-CNRS) à l'Ecole Centrale de Nantes. Ce langage est asynchrone. Il résoud le problème de l'asynchronisme en combinant des modules interruptibles et des modules non interruptibles. Il permet d'associer aux tâches qui prennent un temps physique non négligeable des modules interruptibles. Chaque module peut se trouver dans l'un des trois états : hors-service, prêt ou interrompu. Ainsi, si tous les modules sont déclarés non interruptibles on se ramène à un système strictement synchrone. Un module dit *tâche de fond* qui n'a l'activité que d'attendre une interruption par un événement est définit dans ce langage. Ce module est le seul qui ne se termine jamais.

Ce langage est déterministe en ce sens que la même séquence d'occurrence d'événements conduit à un comportement identique du programme. La compilation d'un programme ELECTRE produit un automate d'états finis. L'asynchronisme augmente le pouvoir d'expression du langage, la contre-partie étant la difficulté de vérification des échéances

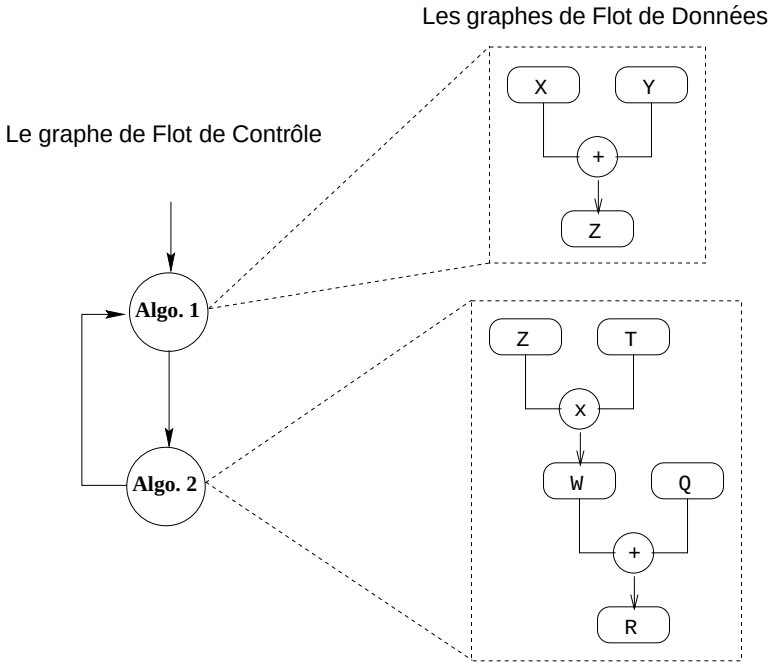


FIG. 2.4-4 – *Les graphes de flot de contrôle et de flot de données.*

temporelles.

Remarques 2.4.1

1. *L'utilisation d'un langage flot de contrôle (resp. flot de données) pour traiter une application où les aspects algorithmiques liés au flot de données sont prépondérants (resp. aspects algorithmiques de contrôle prépondérants) est non adaptée mais possible.*

2. *Comme on l'a déjà vu au premier chapitre, la réalisation d'un contrôleur autonome nécessite d'une part un traitement algorithmique sur les données (flot de données de bas vers le haut) et d'autre part un contrôle sur ce flot (flot de contrôle de haut vers le bas). Cela implique la combinaison des deux langages dans une application de contrôle-commande avec des algorithmes complexes, .*

2.4.4 Programmation hétérogène

Plusieurs travaux sont entrepris pour faire de la programmation hétérogène, c'est à dire un programme combinant un graphe de flot de contrôle et plusieurs graphes de flots de données (voir figure 2.4-4). Cette combinaison permet d'adapter chaque style pour la programmation globale d'un système réactif.

ARGOLUS **: un langage mixte impératif et déclaratif**

Des études sur l'utilisation commune des deux langages synchrones de natures différentes ont été faites au laboratoire Verimag de Grenoble. La structure d'un tel programme

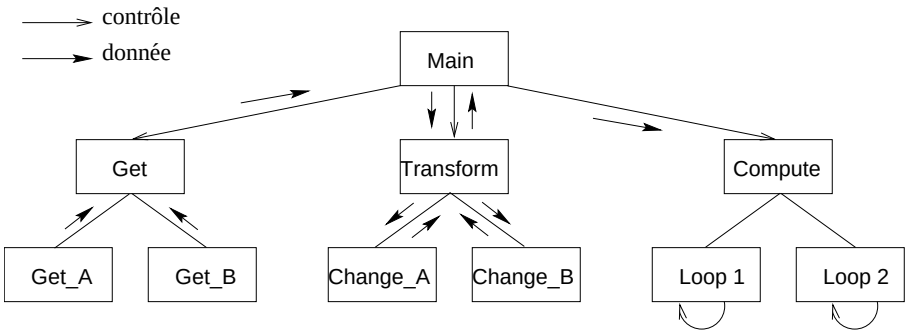


FIG. 2.4-5 – *Un exemple de struct chart.*

est celle d'un programme ARGOS (flot de contrôle) dans lequel on peut remplacer les états de l'automate par des noeuds LUSTRE. Ainsi, l'aspect gestion d'activités et d'interruption est faite en ARGOS, alors que LUSTRE facilite la programmation des actions internes aux activités. Le compilateur ARGOLUS [JLMR94, Jou94], fonctionne en deux étapes :

① Le code LUSTRE est d'abord transformé en une table de transition;

② Le résultat est compilé par le compilateur ARGOS.

La plate-forme SPORTS

Les langages retenus par la plate-forme SPORTS (Synchronous Programming Of Real Time Systems) [ABGM96] sont les langages ESTEREL et LUSTRE et GRAFCET. Le GRAFCET est utilisé pour le contrôle de séquences simultanées, et le langage ESTEREL pour le traitement des préemptions. Les comportements plus combinatoires ou fonctionnels sont programmés par le langage déclaratif LUSTRE. Dans un souci de réutilisation, les classes synchrones associées aux modules synchrones écrits en ESTEREL, LUSTRE ou GRAFCET peuvent être instanciées sous la forme d'objets synchrones interconnectés.

STRUCTURE-CHART

Le modèle d'une structure-chart (figure 2.4-5) est constitué d'un ensemble de noeuds, représentant les activités, et un ensemble d'arcs, qui représente la procédure ou la fonction appelée. Les données qui passent entre les activités sont représentées par les arcs [Gon94].

2.5 Vérification formelle des systèmes réactifs

La vérification formelle des systèmes réactifs est un élément clé pour la sécurité de leur fonctionnement. Les méthodes de vérification sont basées sur l'analyse du modèle du système à vérifier. Généralement, le système est modélisé par un *automate* qui représente l'ensemble des comportements possibles du système. La nature de l'information portée par les états/transitions de l'automate exprime la capacité de pouvoir de vérification.

2.5.1 Les automates booléens

Les automates booléens ne portent que des informations booléennes sur les transitions, ce qui permet grâce au graphe de décision binaire de vérifier toutes les propriétés souhaitées. Malgré le faible pouvoir d'expression de ce modèle, les propriétés de *sûreté* et les propriétés de *vivacité* sont vérifiables. La première propriété exprime le fait qu'une chose mauvaise n'arrivera jamais, la deuxième, que quelque chose de bon arrivera soit sûrement soit probablement.

2.5.2 Les graphes temporisés

Les graphes temporisés sont des automates étendus avec un ensemble de variables réelles, appelées *horloges*, dont les valeurs augmentent uniformément avec le passage du temps.

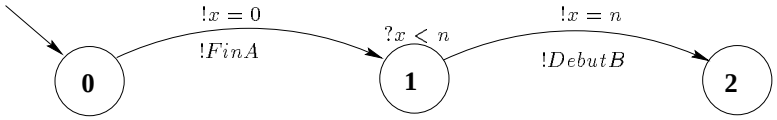
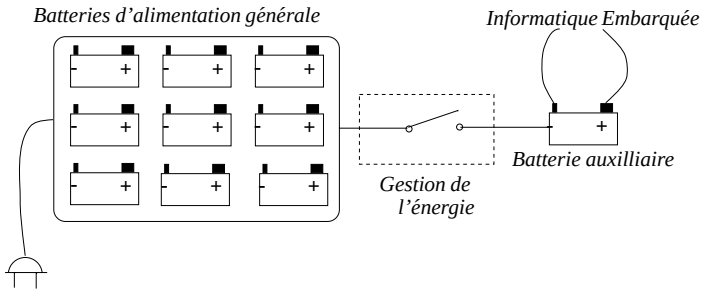


FIG. 2.5-6 – *Exemple de graphe temporisé.*

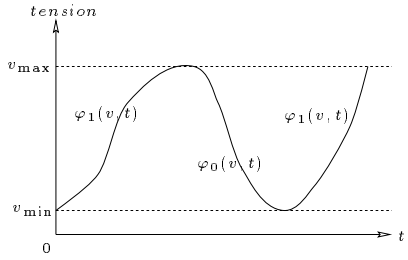
Exemple La figure 2.5-6 donne un exemple de graphe temporisé ayant une horloge x pour mesurer le temps. L'exemple modélise un séquencement de deux tâches **A** et **B** avec la contrainte suivante : la fin de la tâche **A** est séparée par n unités de temps du début de la tâche **B**. L'automate démarre dans l'état 0, avec une valeur quelconque de l'horloge x. L'arc de l'état 0 vers l'état 1 est étiqueté par "FinA". Il est franchi quand "FinA" est vraie. Lorsque cet arc est franchi, ceci signifie que la tâche **A** est finie, l'horloge est remise à zéro. Le système reste à l'état 1 tant que la condition associée à cet état est vérifiée par la valeur de l'horloge : donc le système restera dans l'état 1 pendant exactement n unités de temps. La transition vers l'état 2 est alors franchie immédiatement. Ce qui signifie que le début de l'exécution de la tâche **B** est, effectivement, séparé de la fin de la tâche **A**, par n unités de temps.

Caractéristiques Les horloges qui mémorisent les informations relatives au temps dans les graphes temporisés permettent de mieux modéliser les systèmes réactifs fortement contraints par l'évolution du temps comme les systèmes temps réel. En effet, dans ces systèmes, le temps apparaît comme une contrainte logique de base et non comme un facteur de performance. Deux caractéristiques montrent l'intérêt de ce modèle par rapport au modèle booléen :

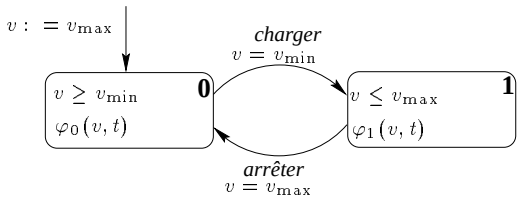
– Pour exprimer la durée, des structures de contrôle à base de compteurs sont utilisées. Avec les automates booléens, ces valeurs sont mémorisées sous forme d'états, d'où le problème d'explosion lorsque plusieurs compteurs évoluent simultanément.



(a) Le dispositif de gestion.



(b) Evolution de la tension.



(c) Le système hybride.

FIG. 2.5-7 – *Exemple d'un modèle hybride.*

Au contraire, dans les automates temporisés, ces valeurs sont mémorisées par des variables.

– Cette apparition de la durée sur les transitions permet de vérifier certaines propriétés temporelles, dont les propriétés *quantitatives temporelles temps réel*. Cette propriété représente en plus de l'ordre d'exécution des actions du programme, l'écoulement du temps entre ces actions.

2.5.3 Les automates hybrides

Les automates hybrides sont des automates qui combinent des composantes *discrètes* et des composantes *continues*. Les variables d'état et le temps évoluent de façon continue et les transitions discrètes peuvent être franchies instantanément. Ces automates représentent en général les systèmes temps réel réactifs, contrôlés dans un environnement qui évolue avec le temps. Nous donnons un exemple d'un système hybride largement inspiré de l'exemple du thermostat donné dans [ACH+95].

Exemple Prenons l'exemple d'une "voiture électrique intelligente". La voiture dispose d'un ensemble de batteries pour rouler et d'une batterie auxiliaire pour alimenter son système informatique. Un dispositif de gestion d'énergie (figure 2.5-7(a)) est mis entre les deux. Il doit maintenir la tension v de la batterie auxiliaire entre une valeur minimale $v_{\min}$ et une valeur maximale $v_{\max}$ (figure 2.5-7(b)). La figure 2.5-7(c) représente le système hybride qui modélise le dispositif.

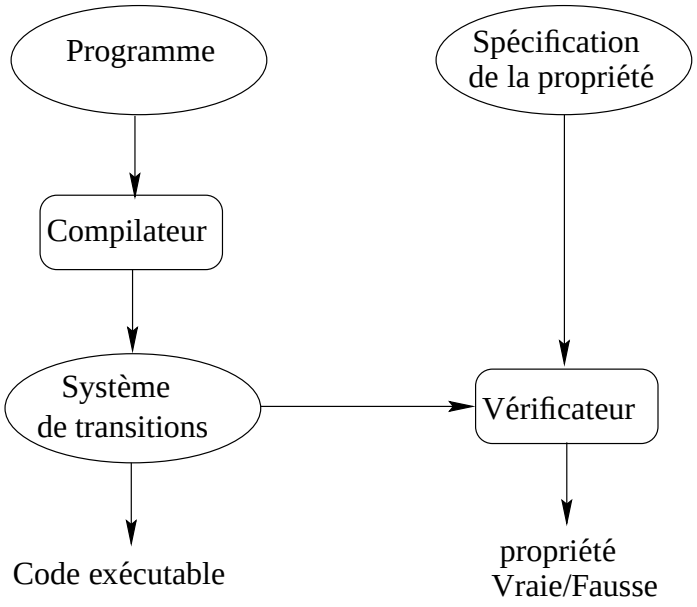


FIG. 2.6-8 – *Schéma de vérification.*

La tension v est mesurée par un voltmètre de façon continue. Le système de gestion de l'énergie qui contrôle cette tension est soit allumé soit éteint :

– Quant le système de gestion est éteint (état 0), la batterie auxiliaire se décharge dans le système informatique et la tension v diminue avec le temps suivant une loi de décharge : $\varphi_0(v,t)$ où t représente le temps;

– Quant le système de gestion est allumé (état 1), la batterie auxiliaire se décharge toujours dans le système informatique et en même temps elle se charge par l'intermédiaire des batteries du bloc d'alimentation générale. Dans ce cas la valeur de la tension doit augmenter suivant une loi de charge : $\varphi_1(v,t)$ où t représente le temps.

La représentation graphique est montrée dans la figure 2.5-7(c). Elle correspond à deux états du système de gestion : éteint (sommet 0) ou allumé (sommet 1). Les transitions entre les sommets sont effectuées quand la tension atteind les valeurs extrêmes : à $v_{\min}$ (le gestionnaire d'énergie *charge* la batterie auxiliaire à partir des batteries d'alimentation générale) et à $v_{\max}$ (le gestionnaire *arrête* la charge).

2.6 Méthodes de vérification

Les programmes vérifiables doivent être exprimés dans un langage qui possède une *sémantique opérationnelle*. Cette sémantique permet d'associer à tout programme un modèle de comportement. La vérification d'une propriété (figure 2.6-8) consiste à :

– donner une description finie du comportement du système réactif. Généralement, le comportement du système est modélisé par un automate d'état fini. D'autres modèles

existent, comme le modèle du système d'équations polynomiales utilisé par SIGALI [Dut92] pour vérifier les programmes SIGNAL. Un programme SIGNAL est compilé en un système d'équations qui manipule des variables représentant les états et les événements. Trois équations sont utilisées pour représenter l'évolution du système : l'équation des contraintes (pour transiter d'un état à un autre); l'équation d'évolution (l'évolution des états) et l'équation de l'état initial. Les entités manipulées sont modulo. 3 : $-1, 0, 1$ pour dire respectivement que le signal est *présent et faux, présent et vrai* ou bien *absent*.

– exprimer formellement la propriété à vérifier;

– utiliser une méthode formelle de vérification. Il existe trois méthodes de vérification, par *simulation comportementale*, à l'aide de *logiques temporelles* et par *la méthode des observateurs*.

2.6.1 Simulation comportementale

La propriété est exprimée sous la forme d'un automate. La vérification consiste à comparer l'automate modèle du système avec celui de la propriété. La comparaison consiste à vérifier que tous les comportements du modèle sont contenus dans l'ensemble des comportements exprimés par la propriété. Intuitivement, l'automate de la propriété contient tous les "bons" comportements possibles du programme vis à vis de cette propriété. De nombreux outils de vérification utilisent cette méthode : AUTO [RdS90] (développé dans l'environnement du langage ESTEREL), ALDÉBARAN [Mou92] (connecté au langage asynchrone des protocoles de communication LOTOS [FGM+92]), CONCURRENCY-WORKBENCH [CPS89].

2.6.2 Utilisation d'une logique temporelle

Au lieu d'exprimer la propriété à vérifier par un automate, on peut la modéliser par une formule d'une logique temporelle. Plusieurs logiques temporelles existent, la plus connue est la logique CTL (Computational Tree Logic). Des algorithmes dits de "model-checking" sont utilisés pour calculer l'ensemble des états qui satisfont la formule. Plusieurs outils de vérification utilisent ce type d'algorithmes, parmi lesquels on peut citer les systèmes EMS [CES83] et XÉSAR [RRSV87].

2.6.3 Méthode des observateurs

Le principe de cette méthode consiste à placer en parallèle avec le programme à vérifier un observateur. En fonction des entrées et sorties du programme, l'observateur décide si le comportement du programme est correct par émission d'une sortie particulière. Ce principe de vérification n'est valable que si l'observateur n'influe pas sur le programme observé. Le mode de communication par diffusion synchrone, utilisé dans l'approche synchrone, permet d'ajouter cet observateur comme composante du programme sans influencer son comportement. Cela permet d'utiliser le compilateur d'un langage synchrone comme outil

de vérification. C'est ce qu'utilise l'outil LÉSAR [HLR93] pour vérifier les programmes LUSTRE.

2.6.4 Model-cheking de TCTL

La vérification sur modèle "model-checking" est utilisée pour vérifier des propriétés *quantitatives temporelles temps réel*. La logique temporelle TCTL (Timed Computational Tree Logic), qui est l'extension de la logique CTL, peut être utilisée pour exprimer les propriétés de tels automates. La méthode de vérification repose seulement sur l'existence d'un algorithme de "model-checking". Deux approches peuvent être distinguées :

L'approche énumérative qui n'est possible que si le modèle est fini. En effet, l'évaluation de la formule est faite après l'énumération de tous les états du modèle.

L'approche symbolique qui peut être appliquée sur des modèles infinis. Les états sont représentés par des prédicats et la formule est évaluée sans construire le modèle. L'outil KRONOS [NSY92, Yov93] utilise cette approche. Cet algorithme a été étendu pour traiter la vérification des propriétés des automates *hybrides*. Par cette extension, la vérification devient *semi-décidable*, c'est à dire décidable avec des conditions sur les automates hybrides et sur les formules de la logique.

2.7 Conclusion

Nous avons présenté dans ce chapitre, les méthodes de spécification des systèmes réactifs, leurs modèles et les techniques formelles de leur vérification. Une analyse des divers approches de spécification nous a montré l'apport des langages synchrones, notamment le formalisme de haut niveau pour décrire les modèles formelles. A l'intérieur de ces langages, on distingue les langages impératifs, et les langages de type déclaratifs. Ces deux modes de programmation sont complémentaires : les langages impératifs sont performants dans les domaines de contrôle de systèmes avec changement d'états (car ils utilisent un formalisme basé sur les automates à états finis), alors que les langages déclaratifs sont performants pour des systèmes où le traitement de flots de données est dominant.

Pour vérifier formellement les systèmes réactifs, il faut utiliser des modèles. Il en existe plusieurs. Le plus simple est l'automate booléen. Avec ce modèle nous pouvons vérifier le comportement logique du système. La simplicité de ce modèle permet d'avoir une réponse rapide à tous ce qu'on souhaite vérifier. Par contre le modèle temporisé permet de vérifier, en plus des comportements du système, le temps qui s'écoule entre ces comportements.

Dans le chapitre suivant, nous présenterons l'environnement de programmation robotique ORCCAD. Dans ce système, les compilateurs des langages synchrones peuvent être utilisés, dans ses différentes couches, pour traduire la spécification en un modèle formelle. Vu que nous intervenons au niveau de la spécification de missions où plusieurs algorithmes de contrôle sont enchaînées, nous utiliserons un langage synchrone impératif. Les expérimentations effectuées à l'aide du langage impératif ESTEREL et des outils AUTO

-simulation comportementale- sont présentées dans les chapitres 3 et 4 de la deuxième partie du document.

Chapitre 3

ORCCAD : Un Environnement de Programmation Robotique de Nature Hybride

NOUS venons de voir, dans le chapitre 1, que l'architecture de contrôle hybride, combinant raisonnement et mécanisme de réactivité, est celle qui donne au robot une évolution performante et que l'approche synchrone est la méthode de spécification qui permet de produire des modèles formels avec un formalisme haut niveau (chapitre 2). De plus les langages synchrones remplissent bien les propriétés exigées par un environnement de programmation robotique (paragraphe 1.4 du chapitre 1). Nous nous intéressons à présent au contrôleur robotique hybride ORCCAD et son environnement de programmation sous lequel cette architecture doit fonctionner.

Nous présentons dans ce chapitre l'environnement ORCCAD tel qu'il est défini dans [SECK93] et dans [Kap94]. ORCCAD est un environnement logiciel permettant de concevoir et de mettre en œuvre le contrôle et la commande d'un système robotique complexe. Il permet également la spécification et la validation des missions à réaliser par ce système. Il est principalement destiné aux applications temps réel critiques en robotique, dans lesquelles les aspects relevant de l'automatique (les asservissements, les commandes) sont amenés à interagir étroitement avec ceux manipulant des événements discrets. Dans cette classe d'applications, ORCCAD s'adresse particulièrement aux systèmes présentant une forte interaction avec l'environnement, par le biais de nombreux capteurs et actionneurs. Le contrôle/commande de ces systèmes est souvent embarqué, et le caractère critique de l'application apparaît dans le coût extraordinairement élevé attaché à une défaillance : l'impossibilité ou la difficulté d'intervention sur un sous-marin autonome à longue portée, sur un engin intervenant après un incident technologique majeur ou sur un véhicule planétaire rendent impératif la minimisation du risque de non réalisation de la mission. A cet effet, ORCCAD offre sûreté de programmation et possibilités de validation par vérification formelle.

Ce chapitre est organisé en trois parties :

- ☞ la section 3.1 présente les principes de l'environnement ORCCAD ;
- ☞ la section 3.2 présente la définition des entités TR et PR utilisées par l'environnement ;
- ☞ la section 3.3 contient la traduction des TRs et des PRs en ESTEREL ;
- ☞ la section 3.4 présente la vérification logique des TRs et des PRs ;
- ☞ l'aspect de l'implémentation et la TM sera traité dans la section 3.5 ;
- ☞ une classification générique des TMs destinée à faciliter la conception des TRs pour les robots mobiles sera donnée dans la section 3.6.

3.1 Principes

La définition formelle de l'action robotique est la formulation utilisée dans ORCCAD [SECK93] pour spécifier, valider par des méthodes formelles et par simulation, et implémenter systématiquement des tâches et des missions robotiques. ORCCAD est basé sur les principes suivants :

- Généralement, les actions physiques que le robot doit exécuter peuvent être ramenées aux problèmes d'automatique et être traitées en temps réel en utilisant des boucles adéquates de commande. Ainsi, la théorie de l'automatique doit être utilisée le plus souvent pour réaliser les actions complexes. Notons que l'approche "fonction de tâche" a été développée pour traiter les tâches référencées capteurs [SBE91] ;
- L'action physique n'est pas suffisante pour définir complètement une action robotique : les instants de départ et d'arrêt doivent être considérés par observation des événements significatifs durant l'exécution de la tâche ;

ORCCAD est un contrôleur de nature hybride à trois couches (figure 3.1-1). Cette structure permet à des utilisateurs de compétences différentes de concevoir des tâches et des missions. Comme dans tout système structuré ORCCAD manipule des entités qui sont les **P***rocédures*-**R***obots* (PR) au niveau contrôle d'exécution, les **T***aches*-**R***obots* (TR) au niveau fonctionnel et les **T***âches*-**M***odules* (TM) pour l'implémentation d'une TR (table 3.1-1).

- ➤ Une PR est une action robotique d'un degré de complexité variable, destinée à réaliser un sous-but, contribuant à la satisfaction de l'objectif global du robot. Ce niveau doit être accessible par l'utilisateur du système robotique.
- ➤ Une TR permet d'implémenter, d'une manière structurée, une loi de commande en combinant un aspect algorithmique (loi de commande contrôlant l'action) et un aspect réactif codé par un automate lié au comportement associé à cette loi (conditions de démarrage, d'arrêt, traitement d'exception,...).

TAB. 3.1-1 – *Entités des systèmes structurés.*

Système	Fonction	Structure	Comportement	Contrôle d'exécution
ORCCAD	Tâche-Robot	Tâche-Modules	ATR	Procédure-Robot
CONTROLSHELL	Modules	Modules-habitats	-	Modules de transition
STATECHARTS	Actions	Activity-charts	Statechart de contrôle	Statechart de contrôle
CHIMERA	Code des port-base objects	Tâches	Graphe d'état	Jobs

➤ Les TMs sont des tâches temps réel pouvant communiquer entre elles par l'intermédiaire de ports typés grâce à des primitives d'un OS temps réel. Un ensemble de TMs constitue une TR.

Pour cela, nous disposons d'un environnement graphique (voire annexe D) et un générateur automatique de code temps réel sous le système d'exploitation VXWORKS.

3.2 Spécification avec les TRs et les PRs

3.2.1 La Tâche-Robot (TR)

La TR est l'entité minimale manipulée par l'utilisateur final et l'entité de taille maximale considérée par l'automaticien. Sa conception est détaillée dans [SECK93].

Définition de la TR

Une TR est la spécification complète et paramétrable d'une action élémentaire d'asservissement, c'est-à-dire de l'activation d'une *loi de commande* de structure invariante sur la durée de la tâche, et du *comportement logique* associé à un ensemble de signaux concernant l'action élémentaire d'asservissement.

La loi de commande

La spécification de la TR consiste à expliciter l'ensemble des fonctions, modèles et paramètres de l'expression analytique, en temps continu, des consignes à appliquer aux actionneurs dans le but d'atteindre l'action physique désirée. En général, la TR est décomposée en modules fonctionnels qui échangent des données. Les caractéristiques liées à l'implémentation (discrétisation, communications, ...) qui complètent la spécification de la loi de commande seront prises en compte par l'entité TM.

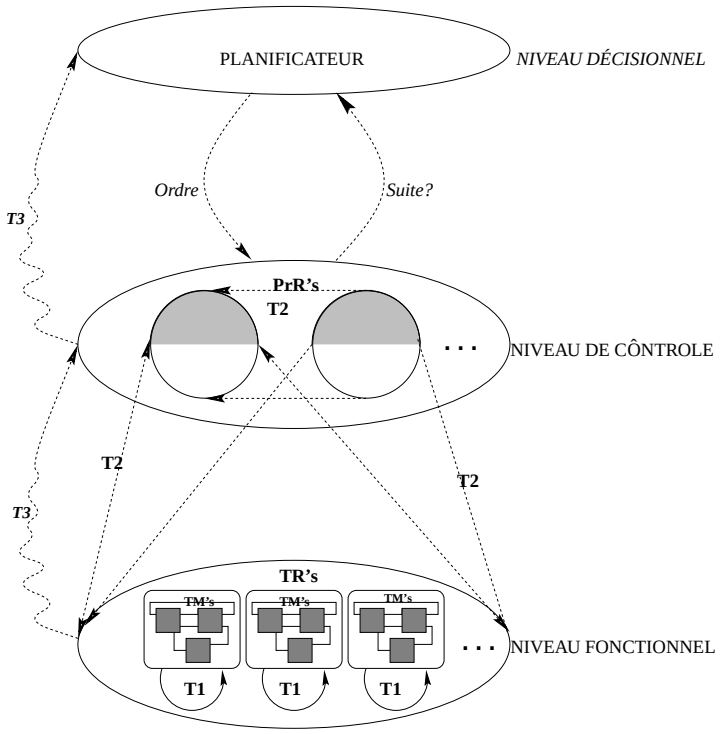


FIG. 3.1-1 – *Les couches dans* ORCCAD.

Le comportement logique

Il représente la spécification logique de la TR en considérant les événements dans le traitement de l'action. Dans le but d'une conception plus aisée et plus systématique les événements sont typés de la manière suivante :

Les pré-conditions : le démarrage de l'asservissement est conditionné par leur présence. Un chien de garde temporel est associé à chaque pré-condition.

Les exceptions : Elles sont générées pendant l'exécution de l'asservissement. Le traitement associé aux exceptions dépend du type de l'exception :

- **Type 1 :** la réaction à ce signal d'exception est interne. Elle consiste par exemple à modifier un ou plusieurs paramètres dans le schéma d'asservissement.
- **Type 2 :** le traitement de l'exception nécessite l'activation d'une nouvelle action. La TR en cours est suspendue (pour une éventuelle ré-utilisation) et le niveau supérieur est informé pour une récupération de l'action.
- **Type 3 :** cette exception correspond à un événement fatal irrécupérable. La TR est tuée, le niveau supérieur est informé pour déclencher une procédure qui ramène le système robotique à un état sûr.

Les post-conditions : leur présence conditionne la bonne terminaison de la TR. Des chiens de garde sont associés à chaque post-condition.

3.2.2 La Procédure-Robot (PR)

L'enrichissement de la TR avec des événements provenant de l'environnement permet de spécifier des actions plus complexes : c'est le rôle de la PR. Le but de cette entité est de définir formellement une action robotique plus complexe qui peut remplir le but assigné par le niveau planificateur. Sa spécification d'une manière structurée par composition logique et temporelle de plusieurs TRs, permet d'atteindre ce but de plusieurs manières différentes en s'adaptant aux conditions des échecs dans l'exécution des TRs.

Définition de la PR

D'après [Kap94], une PR est la spécification complète et paramètrée :

- d'`un programme principal` (l'exécution nominale de l'action), composé d'un arrangement logique et temporel d'un ensemble de TRs, PRs et de conditions ;
- d'un ensemble de `triplets (événement, traitement, assertion)` qui indique le traitement à effectuer dès la réception de l'événement, et l'information à transmettre au niveau planificateur (s'il existe) ;
- d'`un comportement logique` rythmé par les événements qui peuvent être reçus ou émis avant, pendant, ou après exécution.

D'après cette définition, le niveau d'abstraction est élevé par rapport à la TR puisque la PR n'a pas besoin de considérer les aspects liés à la loi de commande. Par contre, comme on va le voir par la suite, il existe une similitude avec la TR dans la description du comportement logique puisque les deux entités représentent une action robotique.

Le programme principal

Le programme principal manipule des TRs et des PRs. La première PR construite ne peut utiliser que des TRs, par la suite elle pourra être utilisée dans le programme principal d'une autre PR. L'utilisateur n'aura qu'à écrire ce programme en utilisant un ensemble d'opérateurs définis dans [CMER92]. Les opérateurs sont classés comme suit : primitives indépendantes de l'environnement, primitives réactives et primitives de synchronisation (tableau 3.2-2).

Le triplet (événement, traitement, assertion)

Les événements sont typés comme dans la TR. On retrouve les pré-conditions, les exceptions (type 2 et type 3), les post-conditions et les signaux internes à la PR. Les traitements associés aux exceptions sont le lancement d'une action robotique par l'intermédiaire d'une PR ou d'une TR à la réception de l'exception de type 2 ou l'arrêt à l'exception de type 3. Le champ assertion est une information (prédicat, paramètre,...) qui passe au niveau planificateur.

TAB. 3.2-2 – *Les opérateurs sur les TRs et les PRs.*

Primitives indépendantes de l'environement	
`Seq(corps,{corps})`	exécution séquentielle
`Par(corps,{corps})`	exécution parallèle
`Cond(condition, corps, corps)`	exécution du premier ou deuxième corps
`Loop(corps)`	répétition de l'exécution du corps
Primitives réactives	
`When(condition,corps)`	Dés que la condition est vraie, le corps est exécuté. Si la condition devient fausse le corps n'est plus exécuté
`WhenEver(condition,corps)`	Dés que la condition est vrai, le corps est exécuté indéfiniment
`Until(condition,corps)`	Le corps est exécuté jusqu'a ce que la condition soit vraie
Primitives de synchronisation	
`Block(point)`	Blocage sur un point
`UnBlock(point)`	Déblocage d'un point

Le comportement logique

Le comportement logique consiste à :

① attendre en parallèle les pré-conditions ;

② exécuter le programme principal, tout en attendant en parallèle les exceptions, les post-conditions et l'état de retour des actions en exécution. Si après un certain temps, fixé par le chien de garde temporel, les pré-conditions ne sont pas satisfaites l'application est arrêtée.

3.3 Traduction de la Spécification en ESTEREL

3.3.1 Traduction de la Partie Réactive de la TR en ESTEREL

L'automate de la TR peut être composé de quatre modules ESTEREL comme le représente la figure 3.3-2. Son fonctionnement consiste :

– à attendre d'abord que toutes les pré-conditions soient satisfaites avant d'exécuter le code de calcul des TMs ;

– puis à attendre en parallèle toutes les exceptions. On remarque que dans le module d'exception de type 1 il y a une boucle, ce qui fait que le traitement est local. Par contre une exception de type 2 ou de type 3 est suivie d'une sortie pour être prise en charge par le niveau supérieur.

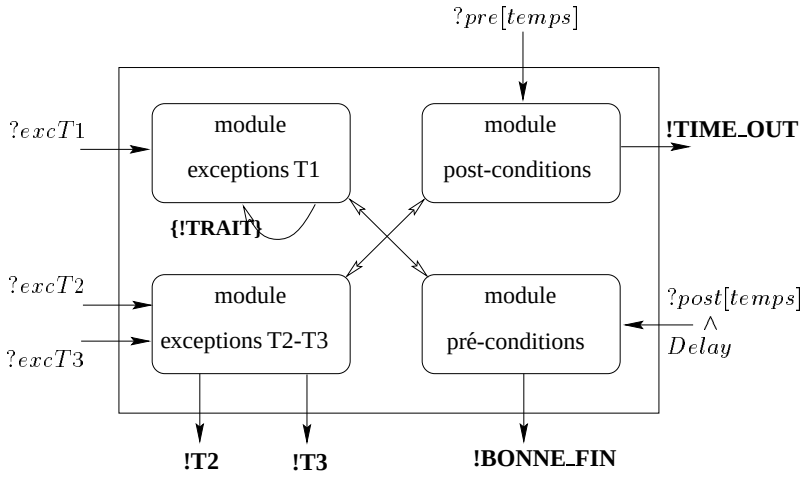


FIG. 3.3-2 – *Les modules* ESTEREL *de l'*ATR.

– en parallèle avec les exceptions, les post-conditions sont attendues aussi. Si une pré-condition est satisfaite l'exécution du code de calcul dans les TMs s'arrête immédiatement et une fonction qui met le système en sécurité est déclenchée.

Voici les modules ESTEREL nécessaires à la spécification de la partie réactive de la TR :

```
% module ESTEREL pour l'attente des preconditions
module PREC :
input VAR_PREC, SECOND;
output VAR_PREC_TEMPS_ECOULE;
constante TEMPS_ATTENTE : integer;
[
do
  await VAR_PREC;
  watching TEMPS_ATTENTE SECOND
          timeout emit VAR_PREC_TEMPS_ECOULE;
end
]
end module

% module ESTEREL pour l'attente des exceptions de type 1
module EXC_T1 :
input VAR_EXC_T1;
output VAR_EXC_TRAIT_T1;
[
loop
   await VAR_EXC_T1;
   emit VAR_EXC_TRAIT_T1;
end
]
```

```
end module

% module ESTEREL pour l'attente des exceptions de type 2 et 3
module EXC_T2_T3 :
input VAR_EXC_T2_T3;
[
await VAR_EXC_T2_T3;
]
end module

% module ESTEREL pour l'attente des post-contions
module POST :
input VAR_POST, DUREE;
output VAR_POST_TEMPS_ECOULE;
constante  TEMPS_ATTENTE : integer;
[
do
  await VAR_POST;
  watching TEMPS_ATTENTE SECOND
  timeout emit VAR_PREC_TEMPS_ECOULE;
end
]
end module
```

3.3.2 Traduction des PRs en ESTEREL

L'architecture du programme ESTEREL produit par la traduction de la spécification des PRs est donnée dans la figure 3.3-3. On notera que cette traduction fait abstraction au signal *assertion* qui lie la PR avec le niveau planificateur. Les détails sur cette traduction sont donnés dans la thèse [Kap94]. Globalement, le programme est composé de quatre modules :

- un module "pré-condition" qui, une fois la demande de `?start` reçue, attend l'ensemble des pré-conditions `?pre[temps]` pour rendre effective cette demande par émission du signal `!START`. Au moment de cette émission tous les modules sont activés. Un chien de garde temporel est associé à chaque pré-condition. Si ce temps est violé, un signal demandant l'arrêt en urgence de l'application est émis `!TIME_OUT`. Notons que fonctionnellement, ce signal est équivalent au signal `!T3` ;

- un module "post-condition". Le rôle de ce module est d'attendre la fin de la PR. De même que dans le module précédent, l'attente devrait être gardée par un chien temporel. La bonne fin de la PR est traduite par l'émission du signal `!BONNE_FIN`. Ce dernier est émis dans deux cas : soit une post-condition est satisfaite, soit la durée de la PR est finie. On notera que la durée n'aura pas de sens lorsqu'une seule pré-condition est spécifiée ;

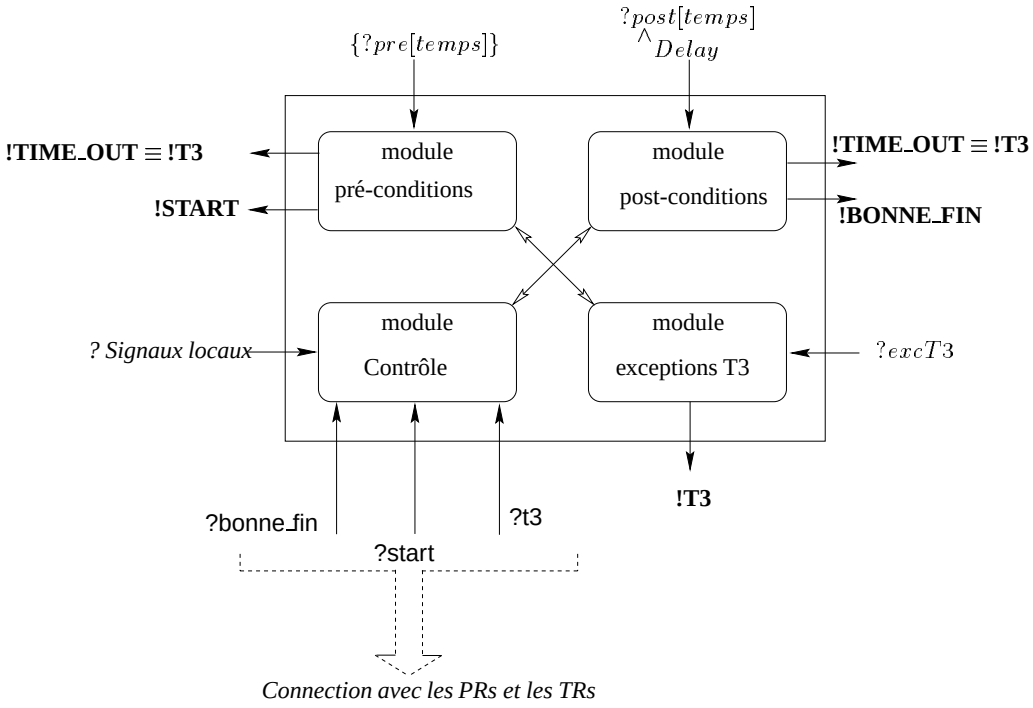


FIG. 3.3-3 – *Les modules d'une PR.*

– un module "exceptions" qui aura la charge d'attendre en parallèle tous les signaux qui peuvent être fatals au déroulement de cette PR ;

– un module "contrôle". Ce module représente l'interface entre cette PR et le reste des PRs et TRs. C'est dans ce module que le corps principal de la PR est écrit en tenant compte des autres modules et des signaux internes.

Une fois qu'une PR est construite, elle peut être considérée comme un module. De l'extérieur, le module est vu avec seulement un signal qui demande son activation avec `?start` et deux signaux qui demandent sa désactivation avec `?t3` ou `?bonne_fin`.

3.4 Vérification formelle sous ORCCAD

La spécification des PRs comporte des signaux d'entrées (dits événements), des signaux de sorties (actions), des chiens de garde temporels et des durées. Bien sûr, une analyse complète devrait tenir compte de ces signaux mais aussi de tous les temps. Le modèle (graphe temporisé) qui en découle est plus difficile et long à vérifier, bien que les propriétés concernées sont certainement plus intéressantes. Cependant, pour accélérer le processus spécification-vérification, un modèle "minimal" qui permet de vérifier le comportement logique de la spécification indépendamment de ces temps est utilisé. Ce modèle (booléen), déjà évoqué, transforme ces temps en signaux logiques.

3.4.1 Vérification logique

Pour rendre systématique le processus de vérification du comportement logique des PRs, il faut :

- automatiser la traduction de la spécification vers un langage qui a une sémantique opérationnelle ; par exemple ESTEREL. Cette traduction doit garder les informations du modèle pour que ce dernier soit vérifiable ;
- classer les propriétés à vérifier et leurs associer la méthode de vérification adéquate. Dans notre cas, on utilisera l'outil AUTO (actuellement MAUTO) qui accepte le format de sortie fourni par le compilateur ESTEREL.

Classification des propriétés logiques Deux classes de propriétés sont identifiées (Tableau 3.4-3). La première concerne les aspects *critiques* comme les propriétés de *sûreté* et de *vivacité* citées précédemment. La deuxième concerne la *cohérence de la spécification avec l'application.*

Vérification logique des TRs

L'outil AUTO est utilisé pour faire la vérification logique. Ce dernier est basé sur la méthode de simulation comportementale. L'automate global est soumis à des opérations telles que : l'abstraction, la réduction,...

La première propriété à vérifier est la propriété de sûreté. Formellement, il faut vérifier : *que tout signal de type 3, spécifié dans la* TR, *est obligatoirement suivi par l'émission d'un signal qui met le robot en position sans risque.*

La deuxième propriété est la propriété de vivacité. Formellement, il faut vérifier que : *le robot a toujours la possibilité d'atteindre l'état "bonne fin" quelque soit l'état où il se trouve.*

Vérification logique des PRs

En plus des propriétés de sûreté et de vivacité, il faut vérifier les propriétés qui dépendent entièrement de la spécification de l'application. C'est à dire, que la spécification est conforme avec le comportement de la PR. Ces propriétés sont classées suivant la relation entre : `action/action`, `événement/action` et `événement/événement` (voir tableau 3.4-3). Par exemple, la première classe permet de vérifier que l'opérateur de "séquencement" entre deux actions se comporte comme il a été spécifié. La deuxième permet de vérifier la réactivité des actions suite à un événement etc...

3.4.2 Autre moyen de vérification

La vérification des propriétés temporelles demande la traduction des spécifications de la TR et de la PR en un langage dont la sémantique permet de donner un modèle temporisé. Un exemple tiré de PRAXITÈLE a été traité dans [Jou95] en utilisant le langage LUSTRE et l'outil KRONOS.

TAB. 3.4-3 – *Vérification des PRs.*

	Propriété Comportemental (ESTEREL/AUTO)	Propriété Temporelle (ARGOS TEMPORISÉ/KRONOS)
Propriétés Critiques	- abscence de deadlock - vivacité - sûreté	- temps d'exécution borné
Cohérence de la spécification	relations : - événement-événement - événement-action - action-action	relations et délai : - événement-événement - événement-action - action-action

3.5 Implémentation de la TR

La spécification fonctionnelle en temps continu des deux composantes de la TR, c'est à dire la loi de commande et le comportement logique doivent être passés en temps discret pour l'exécution sur des calculateurs. Les différentes parties du calcul de la commande et du contrôle seront implémentées sous forme de boucles temps réel grâce aux TMs que nous définirons ci-dessous.

3.5.1 La Tâche-Module (TM)

C'est une tâche temps-réel périodique ou apériodique activée par interruption. Sa spécification prend en considération l'échantillonnage des données, la durée du code de calcul, le type de communication et de synchronisation avec les autres tâches. Sa structure est la suivante :

```
Code d'initialisation;
/* Mise a zero des variables internes,
initialisation des ports de communication... */
while(t<=Temps_TR) { /* tant que la TR n'est pas finie */
Lecture des ports d'entree;
Code de calcul;
Ecriture des ports de sortie;
}
```

Les TMs sont classées en deux grandes catégories en fonction de leur rôle dans l'action robotique :

- ❑ `Ceux qui gérent la partie réflexe de la TR.` Dans cette partie les TMs sont périodiques, et les données échangées décrivent l'état du système. Plusieurs mécanismes de communication et de synchronisation entre les TMs de type Producteur/Consommateur ont été proposés :
  - ➤ `asynchrone-asynchrone` Les deux TMs en communication s'exécutent librement. La TM ne se bloque jamais.

- ➤ `synchrone-synchrone` La TM productrice reste bloqué jusqu'a ce que la seconde soit prête.
- ➤ `asynchrone-synchrone` La TM produit librement et la tâche consommatrice reste bloqué jusqu'a ce qu'une nouvelle donnée soit produite.

Il est évident que suivant le mécanisme utilisé, des données peuvent être perdues. Mais, souvent, il est plus efficace d'accepter de perdre occasionnellement des données, et d'utiliser la dernière donnée afin d'éviter les processus de récupération utilisant de grands tampons. Pour la description des autres types de synchronisation et leur influence sur les performances de la loi de commande nous renvoyons le lecteur à [SFC94].

- ❑ `Ceux qui gérent la partie réactive de la TR.` Les TMs serviront comme interface entre l'environnement asynchrone où les événements arrivent sans aucun ordre préétabli et la partie réactive de la TR. Cette partie est représentée aussi par une TM particulière appelée **A***utomate de la* **T***âche*-**R***obot* (ATR) et un ensemble de TMs appelées *observateurs*. L'ATR n'est pas périodique, elle est activée à la réception d'un signal issu d'un observateur. Le protocole de communication utilisé est de type événement qui garantie l'intégralité des messages et leur mémorisation.

3.6 Classification générique des TMs

Le niveau qui intègre le traitement des asservissements, ou la partie réflexe de la TR , devrait être structuré en entités de traitement (TMs). Cette structuration permet de :

- ❑ faciliter le processus de spécification et de conception des TRs avec un minimum d'effort ;
- ❑ valider la cohérence dans le schéma de contrôle-commande en détectant les incohérences dans la production/consommation des données ;
- ❑ valider que le schéma logiciel spécifié est implémentable sur une architecture matérielle définie ;
- ❑ avoir une modularité dans l'implémentation ;

Contexte informatique La TM est l'entité élémentaire du contrôleur. Ses fonctionnalités sont très proches de l'architecture matérielle. Il nous parait logique de prendre en considération l'architecture pour définir les fonctions "bas niveau" nécessaires au contrôle. L'architecture matérielle est toujours constituée autour d'un calculateur (figure 3.6-4) dont les fonctions principales sont :

- – **L'acquisition des données**. Cette fonction, qui représente la connection entre l'informatique embarquée et les instruments de mesure, est soumise à des contraintes temporelles très strictes. Pour l'exploitation numérique de la mesure, l'instrument est

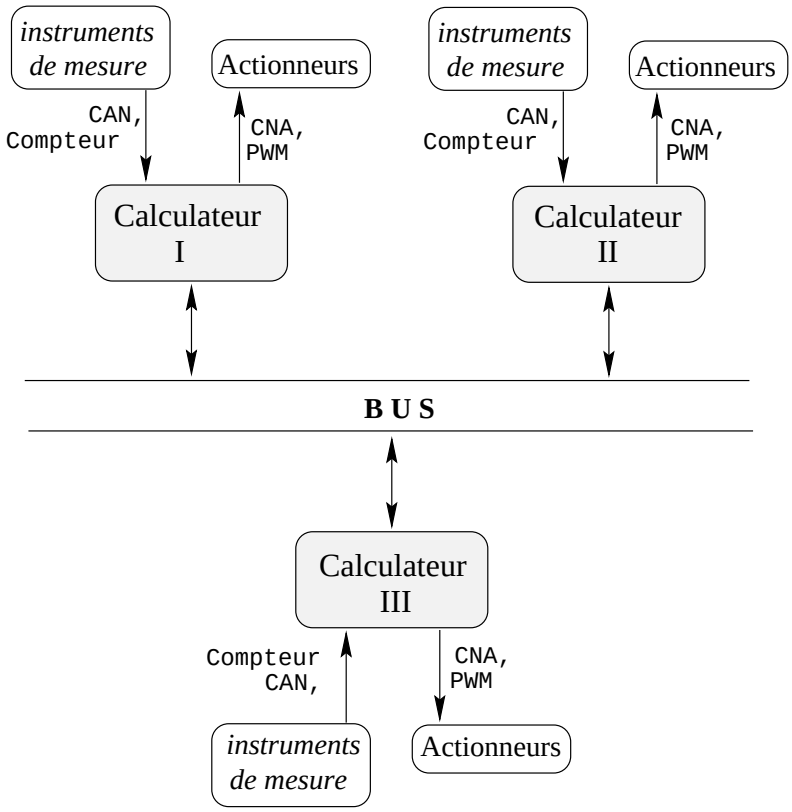


FIG. 3.6-4 – *Architecture matérielle.*

interfacé avec un calculateur numérique (micro-processeur) soit par un Convertisseur Analogique Numérique (C.A.N), soit par une Entrée Tout ou Rien (E.T.O.R), soit par un COMPTEUR. Ces données numériques ou numérisées doivent être disponibles immédiatement pour être utilisées par le calculateur. C'est pour cela qu'il est fréquent d'utiliser un bus local très rapide ou des cartes filles disposées sur le calculateur lui même. Dans ce dernier cas, l'accès aux données se fait directement par mémoire partagée.

– **L'action**. Elle consiste tout simplement à écrire dans un Convertisseur Numérique Analogique (C.N.A) ou à donner un rapport cyclique pour une commande en Pulse Width Modulation (P.W.M). Tout le problème de contrôle-commande consiste à calculer cette valeur à une certaine fréquence d'échantillonnage.

– **La communication** entre calculateurs est nécessaire si leur disposition géographique ou les contraintes temps réel nécessitent l'utilisation de plusieurs processeurs. On trouve au moins un mode de communication sur un calculateur : liaison série (RS232,...), liaison parallèle, bus (VME, VESA, PCI...), réseau (CAN, VAN,...), ... Elles diffèrent par leurs débits et par la distance maximale de communication supportée.

– **Le calcul de la commande**. C'est la fonction de base du calculateur. Le calculateur rassemble les informations des instruments pour produire l'action.

Les fonctions citées ci-dessus interagissent par transfert de données en respectant le cycle : acquisition, calcul, action. L'acquisition et l'action sont deux fonctions (dites : "Drivers") qui dépendent totalement de l'architecture matérielle du système informatique. Par contre, le calcul dépend seulement de la classe des algorithmes à traiter. Nous allons nous placer dans le contexte automatisation d'un véhicule type voiture pour classifier ces algorithmes de calcul.

Contexte automatique Le schéma général d'un asservissement est représenté dans la figure 3.6-5. Le problème de la synthèse d'un asservissement consiste à déterminer un système appelé *correcteur* qui, à partir d'une référence, détermine les commandes u à appliquer pour faire en sorte que les sorties y du *robot* soient aussi proches que possible d'un vecteur d'entrées e. L'entrée peut être représentée par une modélisation de signaux appliqués ou comme la sortie d'un système extérieur ayant effectivement une existence physique. Généralement, le robot est muni d'un système de perception pour se référencer, dans ce cas la commande établie est dite "commande référencée capteurs".

L'implantation directe d'une simple "commande avec retour d'état" n'est possible que si toutes les variables d'état du système sont physiquement accessibles. Dans la pratique, il en va souvent différemment : les informations fournies par les capteurs proprioceptifs forment un vecteur de mesure y, de dimension inférieure à celle de x ; y s'exprime lui-même linéairement en fonction de x et éventuellement de u. Pour implanter la commande, il convient donc de procéder à une reconstruction de l'état x à partir des seules informations dont on dispose, à savoir la commande u et la mesure y. Un élément appelé *estimateur* est utilisé pour faire une reconstruction approchée $\hat{x}$ de l'état x. On trouve également dans la littérature des termes plus ou moins synonymes tels que reconstructeur, observateur, filtre.

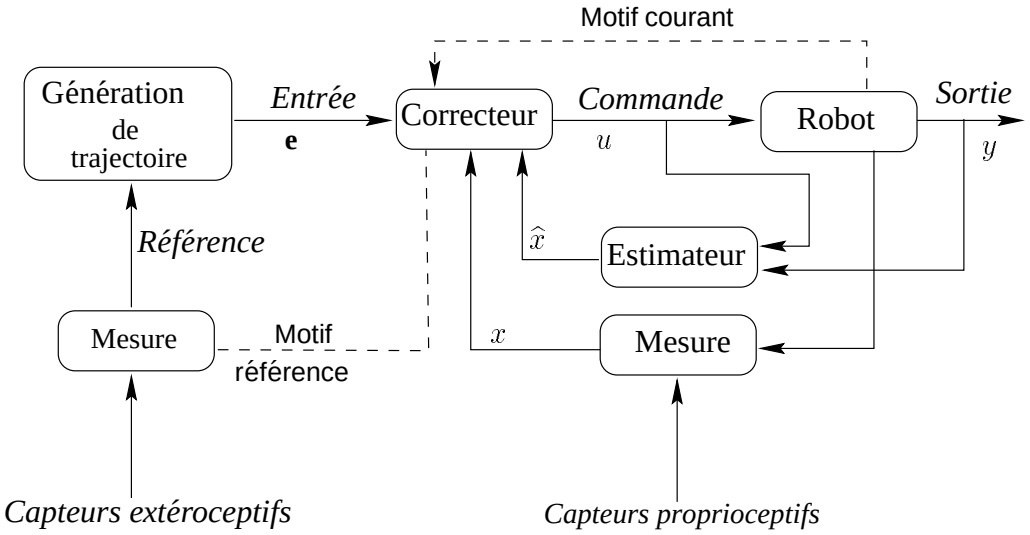


FIG. 3.6-5 – *Commande référencée capteurs.*

Il existe une autre manière de réaliser une commande référencée capteur, et notamment en utilisant la vision. Au lieu de contrôler une situation x entre le capteur extéroceptif et l'objet, on cherche à atteindre une configuration particulière d'un motif dans l'image

(`Motif référence`) en mesurant le `Motif courant`. Ainsi on supprime les erreurs sur l'estimé de la situation x.

3.6.1 Les Classes TMs de la partie réflexe de la TR pour des véhicules type voiture

En tenant compte des contextes informatique et automatique, nous pouvons avoir la classification suivantes (voir figure 3.6-6) :

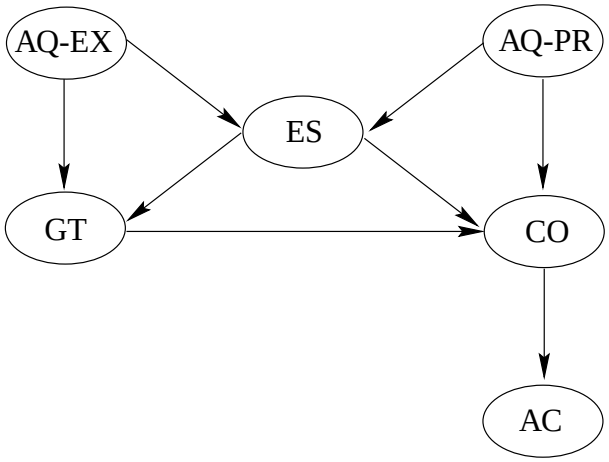


FIG. 3.6-6 – *Relation entre les classes.*

Classe calcul (CA)

C'est la classe qui devrait être la plus indépendante de l'architecture matérielle. A partir d'un vecteur d'erreur obtenu par différence entre le vecteur à réguler et sa valeur réelle (mesurée ou estimée), cette classe calcule la valeur du courant à appliquer aux actionneurs. L'algorithme de calcul peut être compliqué, il peut être lui même composé de plusieurs autres modules distribués sur une architecture multiprocesseurs. Nous pouvons décomposé cette classe en plusieurs sous-classe (figure 3.6-7(a)) :

Classe génération de trajectoire (GT) Elle représente le calcul, à chaque instant, de la consigne qu'on désire satisfaire. La consigne (position, vitesse ou accélération) est soit une valeur fixée à l'avance pour exécuter une trajectoire, soit fournie par la classe AQ-EX (aquisition des capeteurs extéroceptif) pour exécuter une commande référencée capteur.

Classe correcteur (CO) Son but est de calculer la valeur numérique à envoyer vers le C.N.A pour satisfaire une consigne générée. Les données (consignes) reçues proviennent toujours de la classe GT, et de la classe AQ-PR (que nous décrivons par la suite) pour une commande en boucle fermée. Un correcteur type PID est un exemple appartenant à cette classe.

Classe estimateur (ES) L'estimation est nécessaire quand l'information brute mesurée n'est pas suffisante pour établir la loi de commande. La cadence de l'estimateur peut varier énormément. Par exemple, l'estimation d'une pente (ou des frottements) est plus lente que l'estimation de la vitesse de la voiture à partir des données odométriques, puisque la vitesse varie nettement plus rapidement que la pente.

Classe générale (GL) Cette classe est utilisée lorsqu'on a besoin d'un calcul intermédiaire n'appartenant pas aux classes déjà citées.

Classe action (AC)

Généralement pour le contrôle d'un véhicule type voiture, il faut avoir au moins trois actionneurs (voir figure 3.6-7(b)) :

- deux pour le contrôle longitudinal : commande de la locomotion et du freinage des roues ;
- un pour le contrôle latéral : commande du volant de direction ;

Suivant le type d'actionneur utilisé, l'action consiste soit à utiliser un (C.N.A ou une sortie P.W.N. Dans les deux cas, il est préférable d'exprimer les valeurs en pourcentage de la valeur de la sortie maximale autorisée de telle manière à rendre cette fonction l'action indépendante de la configuration de la carte. Par exemple, pour un C.N.A de $12bits$, $12\ Volts$ (c'est à dire $2^{12} = 4048$ correspond à $12Volts$). On écrira 0.5 pour exprimer 50% des $12Volts$.

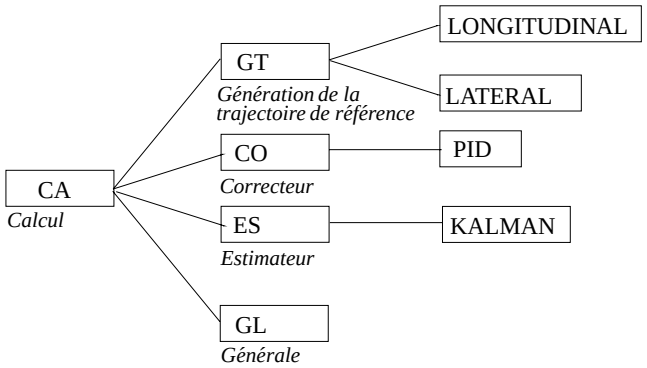
Particularité Cette classe consomme toujours une valeur sans produire.

Classe acquisition (AQ)

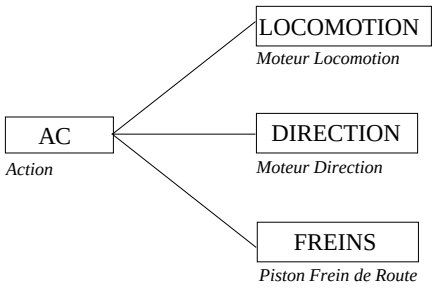
Elle représente les fonctions d'acquisition (figure 3.6-7(c)). Les données peuvent venir soit :

- d'un capteur extéroceptif : caméra, ultra-son, laser... Ces données seront utilisées pour générer la consigne. On appellera cette sous-classe : AQ-EX ;
- d'un capteur proprioceptif : odomètres, vitesse du moteur, angle des roues... Ces données seront utilisées pour avoir l'état interne de la voiture soit directement (données brutes), soit par estimation. C'est la sous-classe : AQ-PR ;

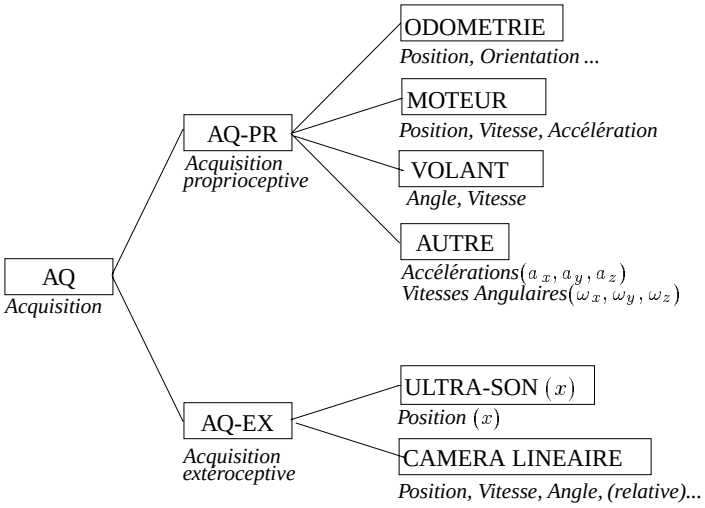
Ces fonctions sont écrites et validées une seule fois. Comme elles sont très proches du matériel, des parties de programmes en assembleur sont parfois utilisées pour rendre efficace l'acquisition. Pour une utilisation directe sans conversion, les sorties de ces programmes devront avoir un sens physique de ce qui est capté. Pour cela, elles devront être exprimées en unité M.K.S.A (Mètre, Kilogramme, Second, Ampère). Un calibrage des capteurs peut alors être nécessaire.



(a) Classe calcul.



(b) Classe action.



(c) Classe acquisition.

FIG. 3.6-7 – *Classification des TMs.*

Suivant la connection entre les calculateurs, la fonction d'acquisition consistent :

– soit à lire directement le contenu d'un C.N.A (mesure d'une vitesse, pression de freinage,...), d'un COMPTEUR (mesure d'une distance ou d'un angle,...) ou d'une E.T.O.R. Si les capteurs en question partagent le même bus de données que le calculateur qui fait l'acquisition ;

– soit à lire, à travers une communication, les valeurs déjà traitées ou acquises par un autre calculateur. Si ce calculateur ne fait que le traitement des mesures (*capteur intelligent)*, la qualité de l'estimation peut être très bonne. Dans ce cas, les paramètres estimés peuvent être considérés comme des mesures dans le schéma d'asservissement.

Particularités

- La classe AQ produit toujours une valeur sans consomme ;
- La sous-classe AQ-EX produit pour la classe GT ;
- La sous-classe AQ-PR produit pour la classe ES.

3.7 Conclusion

Nous avons défini dans ce chapitre l'environnement ORCCAD et ses deux principales entités : la TR et la PR. La première donne à l'utilisateur un moyen de spécification et de conception d'une action robotique simple. La deuxième entité permet de spécifier une combinaison de plusieurs actions pour atteindre un but. Etant donnée qu'elles peuvent être toutes les deux traduites vers le langage synchrone ESTEREL, elles peuvent être vérifiées avec l'outil AUTO.

Pour des raisons techniques d'implémentation, nous avons légèrement changé la traduction (vers ESTEREL) par rapport à celle initialement proposée dans [Kap94]. Plus les programmes traduits regroupent de modules, plus la traduction est simple à mettre en oeuvre en utilisant l'opérateur de mise en parallèle "||"et plus le compilateur produira un modèle complexe.

La traduction modulaire des PRs et leurs connections avec la partie réflexe de l'action constitue un résultat à part entière de notre travail. Les PRs mettent en évidence l'intérêt d'une programmation synchrone en robotique à travers des missions complexes que nous présenterons dans les chapitres 3 et 4. En particulier, il devient possible d'utiliser les outils de vérification des langages synchrones.

Dans le but de minimiser le temps de développement, une bibliothèque de TMs spécifique pour le système robotique ATALANTE sera fournie à l'utilisateur d'ORCCAD. Nous avons commencé par classer les TMs utiles pour la programmation d'un robot mobile à roues de type voiture. Dans le chapitre 2, nous définirons les fonctionnalités de base pour le contrôle-commande des voitures et nous les intégrerons dans ces TMs.

Deuxième partie

Les Applications et la Mise en Œuvre

Chapitre 1

Automatisation des Transports

LA place des transports au sein de notre vie quotidienne est chaque jour plus importante: en particulier, avec l'augmentation de superficie des zones urbaines, la demande en transports professionnels croît sans cesse. Sous la pression de ce facteur, conjugué à la croissance du nombre de voitures par habitant, la congestion et la pollution dans la plupart des villes de par le monde augmentent considérablement depuis quelques années. Dans certains cas, la situation est arrivée à un niveau qui a rendu nécessaire une réglementation de l'usage de l'automobile, et de nombreuses villes sont en train d'élaborer des plans pour dissuader cet usage. Les moyens sont multiples: mesures tarifaires (péages, horodateurs...), réglementaires, ou incitatives (zones piétonnes agréables, transports publics...). Le transport public est actuellement en train de se renouveler et de se donner une image de modernité respectueuse de l'écologie et de la qualité de l'environnement urbain. Économiquement, le transport public se justifie parfaitement dès lors qu'une demande importante de déplacement existe sur un certain axe et sur certaines plage horaires. Dans ce cas, le transport en commun peut apporter un service de qualité grâce à des fréquences élevées, tout en étant particulièrement économe en terme d'utilisation de l'espace par rapport aux débits possibles. En revanche, dès que la demande de transport public descend au-dessous d'un certain seuil, les gains indirects du transport collectif ne compensent plus les dépenses et ce service doit souvent être abandonné. Dans ce contexte, le concept de voitures individuelles publiques en libre service semble pouvoir être le chaînon manquant dans l'offre de transport, entre les solutions actuelles de transport individuel et collectif. Des expériences, dont nous reparlerons, ont été tentées par le passé sans beaucoup de succès; toutefois, les progrès effectués en matière d'automatisation des transports, d'électrification des véhicules, permettent d'espérer de nouveaux développements de ce concept. Le principe commun des projets en cours est d'avoir un très grand nombre de petits véhicules publics à usage individuel (deux à quatre places), en fonctionnement normal sur la voirie traditionnelle et, selon les cas, dans un futur plus éloigné, en fonctionnement automatique sur un réseau propre maillé. Le défi consiste alors à dimensionner cette flotte et à la gérer correctement de manière à pouvoir garantir au moindre coût une grande disponibilité en tout point et à tout moment. L'intérêt économique et écologique de ces systèmes repose sur la combinaison

d'avantages concernant les véhicules: mode de propulsion électrique, petite taille, densité de parking (il n'est besoin que d'avoir accès à la première voiture d'une file), et aussi la réutilisation d'un même véhicule en n'employant qu'une main-d'œuvre réduite.

1.1 Expériences sur le transport public individuel

Nous ne retiendrons ici que les expériences de transport urbain public fondées sur une flotte de petits véhicules urbains motorisés, utilisés à la demande sur des zones diffuses. Nous ne parlerons donc ni des expériences concernant des bicyclettes (Cambridge, Amsterdam ou La Rochelle), ni des systèmes linéaires (cabines sur rails ou câble).

PROCOTIP **Montpellier (1971)** La première expérience en Europe remonte à 1971. Elle repose sur un parc de SIMCA 1000 Automatiques, et sur un dispositif d'accès et de paiement fondé sur l'utilisation d'une clef et de jetons. L'expérience a rapidement échoué à cause de la qualité insuffisante du service; les véhicules étaient souvent indisponibles aux lieux de la demande car en nombre insuffisant et mal gérés. En particulier, aucun moyen n'était prévu pour localiser et redéplacer les véhicules vides (on verra que cette possibilité est l'un des atouts du projet PRAXITÈLE). Aucun organisme n'a voulu assumer les déficits importants des deux premières années.

WITCAR **Amsterdam (1972-1976)** L'expérience d'Amsterdam a fait suite à une première expérience à base de bicyclettes. Elle reposait sur l'utilisation de véhicules électriques conçus spécifiquement pour un usage urbain public et sur des stations de parking où se réglaient les accès aux véhicules grâce à un système de gestion par ordinateur. La première phase prévoyait une flotte de 105 véhicules distribués par 15 stations. En fait, la difficulté d'obtenir des surfaces pour les stations (il semble que la ville n'était pas très favorable à cette expérience) a limité le nombre de stations à 5 et le nombre de véhicules à 35. Dans ce cas et comme à Montpellier, le service (en terme de disponibilité) n'était pas très bon malgré un intérêt certain des usagers et l'expérience a été interrompue.

PGE **Bruxelles (1979)** L'expérience de Bruxelles qui continue encore ne repose que sur 9 véhicules électriques (fabriqués par PGE en Italie) et deux stations. Chaque véhicule a parcouru en moyenne 13000 km sur dix ans avec des trajets moyens de 14 km ce qui fait moins de... 100 voyages par an par véhicule. Là aussi on peut penser que la taille du parc est nettement insuffisante pour un service utile. Nous ne disposons pas de données sur le bilan économique de l'expérience.

MOBILITY ENTREPRISE **Purdue (1983-1986)** Cette expérience sur le campus universitaire de Purdue (USA) consistait à donner à un groupe d'une dizaine de famille l'accès à une flotte de véhicules "publics" c'est à dire possédés et gérés par une association. Chaque famille disposait en permanence d'un petit véhicule urbain (thermique) et avait accès à une flotte de trois véhicules importants avec possibilité de réservation. Cette expérience à durée

limitée a démontré tout l'intérêt des petits véhicules urbains qui remplissaient la majorité des besoins de déplacement (avec 3 places et des déplacements d'au maximum 50 km) et l'intérêt économique du partage d'une flotte pour des déplacements plus importants.

STAR **San-Fransisco (1983-1985)** Ce projet du secteur privé est du même type que le précédent. Il s'adressait à une population d'environ 9000 personnes d'une résidence de San Francisco avec un bon accès aux transports publics. Le principe était celui de la location de voitures à très court terme (facturation à la minute et au kilomètre). 51 voitures de tailles diverses.

MATRA-ARAMIS **Paris (1970-1987)** Contrairement aux projets précédents, le projet ARAMIS se basait sur un transport guidé automatique. Dans le concept initial, les véhicules devaient être de type individuel et se déplaçaient d'une station d'origine à une station de destination sur des voies propres mais sans arrêt intermédiaire. Le réseau pouvait être en anneau simple, en anneau double ou maillé. L'originalité du principe de MATRA était de pouvoir former des trains de véhicules (et d'en sortir) de façon à augmenter notablement le débit du système en cas de besoin. Les problèmes techniques liés à l'automatisation ont pu être résolus par MATRA mais à des coûts qui ont conduit les concepteurs à opter pour des véhicules de plus grande capacité (10 places). A partir de là, le concept devenait moins intéressant, la rentabilité n'était pas assurée et le projet s'est arrêté.

BOEING-PRT **Morgantown** Ce système développé par BOEING est tout à fait similaire au concept ARAMIS mais sans la formation des trains. Ce système est opérationnel à Morgantown (West Virginia) sur une ligne desservant 5 stations mais le concept peut s'étendre à un réseau maillé. Les véhicules (d'une dizaine de places) sont à mi-chemin entre les véhicules individuels et les véhicules en commun, et ne s'arrêtent qu'aux stations nécessaires. Cependant, le coût des infrastructures (voies, stations et dispositifs de gestion) est élevé et compte tenu des débits horaires qui sont nettement inférieurs à ceux d'une ligne de métro, la rentabilité ne doit pas être bonne.

1.1.1 Les projets en cours et futurs

Les projets actuellement en développement présentent des concepts souvent plus réfléchis. Si les programmes ci-dessus ont, pour la plupart, démontré l'intérêt du partage des véhicules, ils se sont souvent limités à un équivalent de location sans service supplémentaire; les projets en cours ont par contre souvent prévu une infrastructure et des dispositifs visant à faire correspondre l'offre de véhicules à la demande.

ENI/AGIP-Italie Ce projet mené en association entre le conglomérat industriel ENI et la compagnie pétrolière italienne AGIP propose de mettre des véhicules électriques (fabriqués en Sicile sous licence LIGIER) en libre service dans les stations AGIP en périphérie des grandes villes (Milan, Rome) avec quelques dizaines de véhicules. Les utilisateurs doivent

rapporter les véhicules empruntés dans la station de départ ce qui apparente le service à un système de location de voiture.

EUROPCAR-La Rochelle Depuis 1994, EUROPCAR en liaison avec la CGFTE (filiale de la Compagnie Générale des Eaux) et la Caisse d'Epargne expérimentent à la Rochelle un principe similaire basé sur des voitures électriques de Renault et de PSA (Clio Electrique et 106 Electrique). Un petit nombre de parkings en libre service serviraient de "distributeurs" de voitures qui devraient être ramenées dans ces parkings en fin d'usage.

VIA TRANSPORT/PSA-Tours Ce projet, aussi fondé sur des voitures électrifiées (AX et 106) est plus sophistiqué puisque les véhicules sont pris pour des petits déplacements et ne sont pas nécessairement ramenés au point de départ. Les problèmes de gestion et de parking deviennent donc beaucoup plus importants, et les solutions sont en cours d'étude.

TAXI 2000-Chicago Après plus de dix ans d'études sur ce concept, les Américains seraient sur le point d'essayer à Chicago et dans d'autres localités un nouveau mode de transport guidé basé sur de très petits véhicules (2 à 4 places) circulant sur une sorte de mono-rail léger et avec de très courtes distances intervéhicules. Les réseaux envisagés sont de type maillé de façon à couvrir de grandes surfaces à demande diffuse.

RUF International En 1991, un inventeur danois qui a depuis fondé une petite société, a déposé un brevet sur un concept intéressant de petit véhicule électrique à deux places circulant en automatique sur un monorail léger et pouvant éventuellement former des trains de véhicules. L'originalité de ces véhicules (en dehors de points de détails sur leur propulsion ou leur freinage) est qu'ils peuvent aussi sortir du rail et circuler sur la chaussée en conduite manuelle. Les véhicules RUF peuvent être des véhicules privés ou publics. Des contacts avec des industriels européens sont en cours pour développer ce projet.

SKY CAB Ce projet d'origine suédoise a acquis en 1992 le statut de projet EUREKA avec une collaboration autrichienne d'un constructeur de petits véhicules pour foires et salons. Le principe est celui de TAXI 2000 : des petits véhicules électriques circulent en automatique sur un réseau maillé sans arrêt dans les stations intermédiaires. Le système serait expérimenté à Gävle (Suède) avec 1300 véhicules et 120 km de voies pour un coût de 2300 MF.

TEMPO La société suisse TEMPO RESEARCH a étudié et fait breveter un système de guidage mécanique pour véhicules privés permettant d'utiliser des voies propres à gabarit réduit. Plusieurs versions de véhicules ont été envisagées allant d'une simple adaptation des véhicules existants à des véhicules électriques entièrement automatisés dans un horizon de 15 ans. Les véhicules adaptés peuvent circuler de façon traditionnelle sur la voirie existante et rentrer sur ces voies propres qui leurs sont réservés et où ils peuvent circuler en pelotons avec accrochage mécanique ou électronique.

AXAR Le professeur Marty, ancien président de l'Institut National Polytechnique de Toulouse à déposé entre 1968 et 1982 un ensemble de brevets sur un système de transport basé sur des petits véhicules électriques circulant en automatique sur site propre. Ce système qui a fait l'objet de nombreuses études pour divers sites est en ce moment considéré pour la desserte du complexe industriel de l'Aérospatiale à Toulouse. Une société (AXAR Transports) a été créée pour étudier et commercialiser ce système.

STATION CAR Un ensemble de villes américaines en association avec des distributeurs d'électricité et des transporteurs locaux, sont en train de monter un consortium pour expérimenter le concept du "station car" ou voiture de gare. Il s'agit d'offrir aux habitants des banlieues un moyen efficace de rallier la gare la plus proche. Ce moyen serait une petite voiture électrique qui serait soit en libre service, soit mis à l'usage exclusif d'une famille par un abonnement ou un crédit bail. L'idée principale est de remplacer la deuxième (ou nième) voiture par un moyen plus économe en énergie, en pollution mais surtout en surface de parking. Des expérimentations de ce concept sont prévues pour 1994, avec vraisemblablement Boston comme ville pilote.

1.2 Le programme de transport public individuel : PRAXITÈLE

La CGFTE (filiale transport publics de la Compagnie Générale des Eaux) et Renault ont signé un accord en 1992 pour développer conjointement un véhicule électrique à usage spécifiquement urbain et public. De son coté, la CGFTE travaille depuis 1989 sur le concept des véhicules urbains en libre service. En 1993, un accord de collaboration a été établi entre la CGFTE (qui représente plusieurs filiales de la Compagnie Générale des Eaux), Renault, EDF, Dassault Automatismes et Télécommunications et deux instituts de recherche publics: l'INRIA et l'INRETS pour lancer un programme baptisé PRAXITÈLE. L'objectif est d'arriver rapidement à des expérimentations pour valider l'intérêt du système auprès des utilisateurs mais aussi de préparer l'avenir en développant les technologies d'analyse des futurs systèmes et en lançant des recherches sur leur évolution technologique, en particulier pour préparer les futures versions qui devraient être plus performantes et apporter de meilleurs services. Les développements techniques concernent l'ensemble des points organisationnels et réglementaires qu'il est nécessaire d'étudier pour arriver à des expérimentations en vraie grandeur du système de transport urbain public individuel dans ses différentes versions. Ces versions sont définies dans [PT93], chaque version inclut les fonctionnalités de la version précédente :

Version 1

Les véhicules sont en conduite manuelle par les usagers ou éventuellement par des conducteurs professionnels (fonction taxi), occasionnels ou permanents.

Le retour à vide des véhicules est assuré par du personnel qui déplace les véhicules sur des remorques ou en formant des trains de voitures avec accrochage mécanique. Chaque véhicule est localisé par le système de gestion qui commande les déplacements à vide en fonction de la demande client anticipée. L'accès, le paiement et la réservation se font par carte intelligente. Les voitures sont garées dans des parkings réservés où elles sont rechargées automatiquement. Des parkings automatiques peuvent être utilisés. Le co-voiturage peut éventuellement être instauré pour réduire la congestion.

Version 2

Les véhicules peuvent se déplacer en pelotons avec accrochage électronique.

La voiture de tête du peloton est conduite manuellement. Les voitures suiveuses sont en conduite automatique. Les pelotons utilisent de préférence des voies propres ou des couloirs réservés. La rentrée et la sortie d'un peloton se fait en automatique dans des stations équipées pour cela et qui font office de parking. Cette version apporte des débits importants et une automatisation de la conduite pendant les périodes de pointe sur certaines voies. L'accrochage automatique immatériel simplifie la formation des trains pour le retour à vide des véhicules. La mise en parking est assurée automatiquement et enclenche la recharge des batteries. Le déplacement dans les grands parkings est automatisé.

Version 3

Les véhicules se déplacent en automatique individuellement ou en pelotons sur des sites propres protégés et équipés.

La rentrée et la sortie des sites propres se fait dans des stations. Ces stations qui font aussi office de parking peuvent s'échanger des véhicules vides sans intervention humaine sur commande du système de gestion. Cette version apporte une automatisation et un service instantané sur les sites propres en permanence. Elle résout aussi le problème du déplacement automatique des véhicules vides entre les parkings qui sont sur le site propre (il peut y avoir d'autres parkings).

Version 4

Les véhicules peuvent se déplacer automatiquement (à vitesse réduite) sur une voie publique légèrement aménagée. Les usagers peuvent choisir une conduite manuelle ou une

conduite automatique sur ces sites. Cette version permet le déplacement en automatique sur tout le réseau et en point à point. Les voitures peuvent être livrées à domicile et rapportées aux parkings après une course quelconque. Cette version très futuriste n'est envisageable que dans le cadre d'une modification profonde du partage de la voirie avec les véhicules privés.

1.2.1 Les thèmes de recherches dans PRAXITÈLE

Pour développer ces versions, plusieurs axes de recherche doivent être explorés :

Thème 1 : conception du véhicule de base

Il ne s'agit pas là de recherches sur le véhicule électrique mais l'adaptation des techniques existantes (robotique, design) au problème du véhicule public urbain. Le cahier des charges doit spécifier les matériaux intérieurs et extérieurs pouvant satisfaire à un usage public, les méthodes de nettoyage, la conception du poste de conduite, l'habitabilité, les accès et les dispositifs mécaniques d'accrochage et de guidage en parking.

Thème 2 : conception des stations de parking

Il y aura deux types de parkings réservés aux véhicules publics : des stations de "bord de rue", vraisemblablement en épi pour économiser de la place, et des dépôts de grande capacité où le déplacement des véhicules devra se faire en automatique avec guidage mécanique (par exemple sur rail central). Dans les deux cas, il faudra étudier les dispositifs de transfert d'énergie en vue du rechargement des batteries sans intervention humaine. Les stations de nettoyage (si possible automatiques) devront aussi être étudiées. Les stations de parking devront comprendre des bornes d'information pour les usagers.

Thème 3 : ingénierie système

Pour avoir un système offrant un bon service et une bonne rentabilité, il est critique de dimensionner correctement certains paramètres : nombre de véhicules, nombre d'emplacement et dimension des parkings, voie propres, liaison avec les autres modes. Nous pensons que des outils performants d'évaluation des performances doivent être développés. Ces outils pourraient être incorporés dans les stations de travail des planificateurs urbains qui disposeraient de toutes les informations concernant leur ville, en particulier grâce aux bases de données géographiques.

Thème 4 : localisation

Pour assurer une bonne gestion du parc de véhicules, il est nécessaire de les localiser. Cette localisation peut se faire à bord du véhicule (GPS, triangulation ou navigation) ou de l'extérieur (triangulation), mais il serait souhaitable que chaque véhicule connaisse sa position pour vérifier les droits de circulation des usagers, réguler la vitesse de pointe, autoriser ou non le stationnement et donner des informations de guidage.

Thème 5 : communication

La communication est nécessaire si ce sont les véhicules qui se localisent et si le contrôle d'accès se fait aussi par le véhicule (et non au dans le parking). Elle doit au minimum se faire au niveau de chaque parking pour que le véhicule communique son état. Un lien permanent est souhaitable pour des problèmes de sécurité et d'assistance à l'usager. La communication sera de type numérique mais devra permettre la transmission de la parole et éventuellement de la vidéo.

Thème 6 : gestion informatique

Pour bien fonctionner, un tel système doit gérer correctement ses ressources. A partir des informations disponibles sur les véhicules (état, localisation) et des demandes de déplacement connues ou estimées, il sera vraisemblablement nécessaire d'effectuer des mouvements à vide. Le système de gestion doit commander ces déplacements à vide en fonction des possibilités de les effectuer (disponibilité du personnel ou de moyens automatiques, dimension des trains, état des véhicules, etc...). Le système informatique doit aussi gérer les accès aux véhicules, les réservations et la facturation.

Thème 7 : organisation, réglementations

Ce thème concerne les opérateurs de réseaux de transport, les collectivités locales, les pouvoirs publics et les usagers. On devra y aborder les problèmes de réglementation (responsabilité, autorisations de conduite, conduite en train, conduite automatique), les autorisations de construire parkings et voies propres, les relations avec les autres transporteurs et particulierement les compagnies de taxi, l'intégration tarifaire, etc...

Thème 8 : automatisation du parking

Que ce soit pour les parkings de bord de rue ou les dépôts, il est souhaitable d'y automatiser les déplacements de véhicules. Cela est particulièrement vrai pour les dépôts où l'emplacement du point de dépose ou du point de prise peuvent être fort éloignés des places de stockage. Il a été reconnu que l'un des freins majeurs à l'utilisation de la voiture individuelle était la difficulté du parking. Il est donc important de simplifier celui-ci au maximum.

Thème 9 : conduite en pelotons

La formation des pelotons peut se faire derrière un véhicule de tête qui serait par exemple conduit par un professionnel sur un circuit bien identifié et à des horaires connus. L'entrée et la sortie de ces pelotons se feraient dans des "stations" qui recevraient un certain équipement au sol pour faciliter la manœuvre automatique. La conduite automatique des véhicules suiveurs concerne aussi bien le guidage longitudinal (maintien des interdistances) que le guidage latéral. En dehors des stations, la voirie ne serait pas instrumentée. Des communications entre les véhicules et avec les stations sont nécessaires.

Thème 10 : conduite automatique sur voie propre

Les procédures de guidage latéral et longitudinal pour la voiture de tête (ou la voiture isolée) doivent être développées en utilisant des dispositifs liés à la voie propre. Les collisions ne peuvent se produire qu'avec des véhicules similaires circulant sur la même voie ce qui simplifie le problème de la détection d'obstacle.

Thème 11 : conduite automatique hors voie propre

Les procédures de guidage ne doivent plus dépendre d'une infrastructure lourde au sol. Les collisions avec des obstacles fixes ou mobiles inconnus sont possibles. Il faut donc développer des dispositifs de détection à bord du véhicule et des procédures d'évitement. De plus, une recherche doit être menée sur le type de voies et à quelle vitesse un tel mode de conduite serait possible.

1.3 Etat de l'art de la recherche sur les thèmes 9, 10 et 11

Notre domaine de travail se situe à lintérieur des thèmes 8 et 9, puisqu'il s'agit de spécifier et d'implémenter les applications de conduite en pelotons et l'automatisation du parking. Cet axe de recherche permet le passage de la version 1 à la version 2, et permet aussi de produire une base technologique pour la version 3 (thèmes 10 et 11).

1.3.1 Historique de la conduite automatique

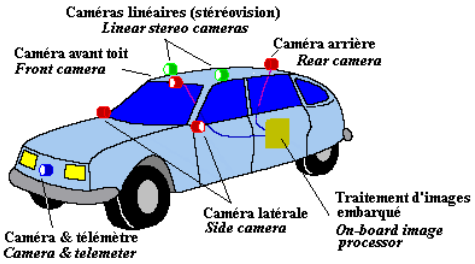
A la fin des années 50, la société américaine GM (**G**eneral **M**otors) a réalisée le premier contrôle automatique d'une voiture [Gar60]. Le système de contrôle était entièrement électronique, d'où son nom "Electronic Highway". Dix ans après, vers 1960, un centre de recherche universitaire de l'OSU (**O**hio **S**tat **U**niversity) a commencé à s'intéresser aux problèmes de conduite automatique des voitures [Bar62]. Des recherches fondamentales à long terme ont été entamées dans cette université [Bar62] entre 1965 et 1980, sur les contrôles (longitudinal et latéral). Les premières applications des technologies de l'automatique dans le transport sont apparues dans le projet METRAN au MIT (**M**assachusetts **I**nstitute of **T**echnologie) en 1966 [Han66].

Suite aux travaux réalisés dans l'OSU, la FHWA (**F**ederal **H**igh**W**ay **A**dministration) a sponsorisé des études sur la faisabilité d'un système de transport automatique. Ces études se sont soldées par deux publications, en 1969 [trw69] et en 1977 [Cor77], traçant de nouveaux axes de recherches. Trois ans plus tard, entre 1980 et 1982, les résultats de ces études ont été repris par GM [Ben81, Ben91]. Le constructeur américain s'intéressait à l'évaluation des techniques pour la réalisation d'une autoroute automatique. Ces travaux, confidentiels, ne sont généralement pas publiés. Et il est difficile d'estimer leur étendus. Certainement GM est l'industriel le plus actif et le plus ancien dans ce domaine. Une petite quantité de ses recherches est disponible pour la communauté des chercheurs dans [SH79].

Une grande partie du travail réalisé dans le domaine de l'automatisation de l'autoroute est actuellement mis en évidence dans les systèmes AGT (**A**utomated **G**uideway **T**ransit)

et PRT (**P**ersonal **R**apid **T**ransit). L'UMTA (**U**rban **M**ass **T**rans portation **A**dministration) les a sponsorisé durant les années 70. Les chercheurs de l'AGT ont trouvé des solutions à la plus part des problèmes posés par l'AVCS (**A**dvanced **V**ehicle **C**ontrol **S**ystems). Ils ont développé des véhicules pouvant opérer en mode manuel ou en mode automatique, et leurs résultats de recherches sont diffusés sur de très nombreux rapports, journaux et conférences. Les références les plus importantes sont les quatre conférences [And72, And74, Gar76, Int78].

Des développements similaires ont commencé en Europe et au Japon vers 1960. En Angleterre, le RRL (**R**oad **R**esearch **L**aboratory) a fait une expérimentation de conduite automatique vers la fin de 1960, tandis qu'au Japon le MEL (**M**echa- nical **E**ngineering **L**aboratory) du ministère MITI (**M**inistry of **I**nternational **T**rade and **I**ndustry) a réussi, dés 1967, à contrôler une voiture à une vitesse de 100Km/h en suivant un câble magnétique [Osh65]. Le programme IVS (**I**ntelligent **V**ehicle **S**ystems) du MITI a aussi réalisé en 1977 une conduite automatique à 30 Km/h en utilisant un système de vision [FYT93]. Durant les années 70, les applications de l'automatisation des automobiles ont été éclipsées par les activités de l'AGT (**A**utomated **G**uideway **T**ransit), principalement le système de contrôle longitudinal des "Cabinentaxi" développé en Allemagne [Hes72], le système CVS au Japon [Ish74] et le système ARAMIS en France [Mau74].



LE VEHICULE DEMONSTRATEUR PROLAB2
PROLAB2 DEMONSTRATOR VEHICLE

L'histoire moderne de l'AVCS a commencé en 1986 parallèlement en Europe et aux USA. En Europe, avec le programme PROMETHEUS (**PRO**gra**M** for **E**uropean **T**raffic with **H**ighest **E**fficiency and **U**nprecedented **S**afety). Ce programme a été lancé à l'initiative des industriels de l'automobile, avec le support du programme EURAKA de la communauté Européenne et des gouvernements des pays de l'Europe de l'ouest. Son but était de fournir une structure de coopération pour les pré-développements des technologies industrielles. Les développements visent à augmenter la sécurité des transports routiers et permettre à l'industrie automobile Européenne d'acquérir une réputation vis à vis de ses homologues des autres continents.

Pendant huit ans, ce programme à investit environ 650 MECU[1] dans les divers projets de recherches dans les domaines des capteurs, des méthodes de fabrication des puces et les technologies de l'intelligence artificielle jusqu'au développement de prototypes de démonstrations.

Les programmes de recherche Européens ont produits plusieurs concepts, proche de l'AVCS, comme la "coopération pour la conduite" et le "Copilotage". La coopération pour la conduite, comme elle est décrite par *Broqua et al.* [BLMM91] consiste à :

– faire un contrôle intelligent pour maintenir un espace longitudinal entre les véhicules et signaler au conducteur les franchissements de la distance de sécurité;

1. 1 ECU=6.5 FF

– faire des manœuvres intelligentes lors des changements de voie et lors des dépassements;

– transmettre en aval les informations au conducteur en cas de problèmes sur la route suite à des accidents, de grave encombrement, ou bien la présence de brouillard ou de verglas.

La référence [BLMM91] décrit, en détail, le concept de contrôle intelligent, avec des courbes qui montrent la performance des véhicules équipés avec ce système et son impact sur les conditions du trafic.

Le concept du "Copilotage" est destiné plus à l'avertissement et à l'information du conducteur. L'acquisition des données et leur interprétation ont pour but de fournir des recommandations au conducteur dans certaines situations à risques. Les implémentations on été réalisées sur le prototype "Prolab2". Sa description est donnée dans [BDR94].

Indépendamment de l'initiative Européenne, le département de Transport de Californie (Caltrans) décide en 1986 d'organiser le développement de nouvelles technologies, économiquement réalisables, pour améliorer la capacité des autoroutes de la Californie. Ceci dans le but d'anticiper les futurs besoins des autoroutes Californiennes. Le "Caltrans" et l'ITSUC (**I**nstitute of **T**rans- portation **S**tudies of the **U**niversity of **C**alifornia) à Berkeley ont créé le programme PATH (pour **P**rogram on **A**dvanced **T**echnology for the **H**ighway, et après **P**artners for **A**dvanced **T**ransit and **H**ighways). Le programme avait trois objectifs : une technologie de propulsion propre, une autoroute automatique et un contrôle automatique. Reconnaissant que la réalisation d'un tel système nécessite un effort national, le "Caltrans" et les partenaires universitaires du projet PATH ont encouragé leurs équivalents des autres états à la création d'un programme national "Mobility 2000" [Sax93, SBP93].

Les ministères du gouvernement Japonais sponsorisent actuellement plusieurs projets :

– MITI (**M**inistry of **I**nternational **T**rade and **I**ndustry) sponsorise les projets PVS (**P**ersonal **V**ehicle **S**ystem) et SSVS (**S**uper-**S**mart **V**ehicle **S**ystem). Ces projets ont pour but la réalisation d'une conduite entièrement automatique. Des tests ont été réalisés avec une camionnette dans une piste d'essai en utilisant une machine de vision et des capteurs ultrasonores pour la détection de la position par rapport à des marquages au sol. Le projet SSVS, comme il est décrit par *Tsugawa et al.* [TWF91], est bien avancé dans les études du concept et l'évaluation mais les développements des prototypes et les validations expérimentales ne sont pas prêts. Ce ministère s'intéresse aussi à l'industrie des automobiles autonomes.

– Le Ministère de Transport a développé le concept ASV (**A**dvanced **S**afety **V**ehicle), qui traite les problèmes de la sécurité dans le transport tel que l'évitement d'obstacles et les systèmes d'aide à la conduite des personnes âgés.

– Le Ministère de la Construction, prend part au projet ARTS (**A**dvanced **R**oad **T**ransportation **S**ystem), pour concevoir le SEAD 21 (**S**uper **E**xpressway for **A**utomated **D**riving - 21) [FYT93]. Ce projet consiste à réaliser une autoroute entièrement automatique, pour le transport des cargaisons dans des tunnels conçus spécialement sous les rues des grandes villes Japonaises.

– Le Ministère de la Construction, et par l'intermédiaire de son institut de recherche le PWRI (**P**ublic **W**orks **R**esearch **I**nstitute) a aussi proposé le système RVCDS (**R**oad **V**ehicle **C**ooperative **D**riving **S**ystem) qui est décrit par *Nakamura et al.* dans [NKYA94]. Le projet s'étale sur plusieurs étapes : au début avec un système de communication entre les véhicules pour la prévention et la sécurité et après progressivement vers l'évitement d'obstacles et éventuellement la conduite automatique. Ce ministère s'intéresse à la construction des infrastructures, en insistant sur les aspects de la communication.

Aux Etats Unis, le développement des systèmes d'assistance et de prévention sont réalisés par les entreprises privés. La NHTSA (**N**ational **H**ighway **T**raffic **S**afety **A**dministration) le département des transports aux Etats Unis supporte le développement des systèmes permettant de déterminer la cause des accidents [LB94]. Durant la période de 1986 à 1993, le développement de la conduite automatique au USA a été entièrement poursuivi par le programme PATH de Californie [Shl92, CHZ+93]. Mais en 1994, la FHWA (**F**ederal **H**igh**W**ay **A**dministration) du département de transports des Etats Unis a lancé le programme AHS (**A**utomated **H**igway **S**ystem) sur les autoroutes automatiques [SB94]. Une analyse des systèmes testés entre 1993-94 peut être trouvée dans [BS94].

1.3.2 Etat de l'art de la recherche sur le contrôle latéral

Début des activités

La conduite automatique des véhicules a commencé sur des maquettes en 1953, ensuite sur des vraies véhicules à partir de 1957 [Ins68]. Les premières voitures automatiques ont été réalisées, dés les années 50, par l'industriel américain GM, suite à une série d'expérimentations à Princeton, NJ [Gar60, BC58, Pro58, FGM+62]. La référence utilisée par GM, pour établir la commande, est un fil électrique installé au milieu de la route et parcouru par un courant de 2 kHz. Une bobine montée dans le véhicule permet de le détecter. Il n'y a pas de document sur la loi de commande utilisée, mais elle devrait sûrement être empirique pour démontrer la faisabilité du concept, et utilisant des circuits analogiques.

Des efforts analogues ont été fournit durant les années 60 respectivement en Angleterre dans le laboratoire RRL (**R**oad **R**esearch **L**aboratory), aux USA à l'OSU (**O**hio **S**tat **U**niver- sity) et au Japon dans le laboratoire MEL (**M**echa- nical **E**ngineering **L**aboratory) du ministère MITI (**M**inistry of **I**nternational **T**rade and **I**ndustry). Le RRL a implémenté des lois de commandes sur deux voitures : une Citroën DS équipée d'un moteur hydraulique sur la direction [Car70]; et une Ford Cortina avec un moteur électrique [Pen]. Les deux systèmes ont posé des problèmes. RRL a apporté une forte contribution au domaine du contrôle latéral, par la réalisation d'un câble de référence sur 16 Km d'autoroute en Angleterre [Res72].

Recherche de l'OSU (Ohio Stat Univer- sity)

L'OSU (**O**hio **S**tate **U**niversity) s'est intéressé à l'automatique du contrôle latéral entre 1960 et 1970, par l'équipe du Professeur *Robert Fenton*. Cet intérêt s'est traduit par la

publication d'une douzaine de rapports. Dans [FM91], on trouve un état de l'art de ces recherches avec plus de quarante références. Les chercheurs de l'OSU ont utilisé les méthodes classiques de l'automatique linéaire (plut tard de l'automatique non-linéaire) pour réaliser les lois de commandes. Ils ont publié en détail une première étude théorique du contrôle automatique pour la conduite des véhicules [Bar62], avec une comparaison des caractéristiques fréquentielles et des réponses temporelles de quatre contrôleurs différents. Les expérimentations ont été faites en utilisant soit un fil, soit deux fils électriques placés sur les trottoirs [FOB71, Ols69, Ols77].

Entre 1965 et 1969, le groupe de recherche de l'OSU a fait ses expériences dans une autoroute en utilisant une voiture de modèle Plymouth, et en atteignant la vitesse de $36\ m/s$. Les manipulations réalisées étaient douce en virage ($0.1\ g$ en accélération latérale) avec une erreur maximale de $2.5\ cm$ dans les conditions normales et $3.8\ cm$ avec une rafale de vent [FMO76]. Ce contrôleur était basé sur les méthodes classiques de la commande des systèmes linéaires avec un modèle dynamique à deux degrés de libertés.

Après, l'utilisation du fil électrique a été abandonnée parce que sa maintenance revenait trop cher [FM91]. Ensuite, *Fenton* et *Selim* [FS88] réalisèrent un premier contrôleur optimal utilisant un observateur qui reconstruisait l'état de la boucle d'asservissement seulement à partir de la mesure de la distance latérale. Leur contrôleur tenait compte de la dynamique de la vitesse latérale. La réalisation a été faite en utilisant un ordinateur analogique.

Autres recherches entre 1960 et 1970

L'approche adopté par le *Dr Sadayuki Tsugawa* et ses collègues du laboratoire MEL (**M**echanical **E**ngineering **L**aboratory) du ministère MITI (**M**inistry of **I**nternational **T**rade and **I**ndustry) au Japon a commencé par utiliser la technique du filo-guidage, entre les années 60 et les années 70. Mais, l'approche fut abandonnée aux profit des techniques de la vision, avec des caméras vidéo montées dans la voiture. Des algorithmes de traitement d'image ont été utilisés pour identifier et localiser la position de la voiture par rapport au lignes blanches [TYHM79, Tsu93].

Un nouveau concept, développé par *Spangler* et *Snell* [SS62] et illustré dans le système TRW [trw69], utilise un émetteur. Ce dernier, placé dans le véhicule, provoque un courant dans le conducteur passif placé sous la route. Une bobine permet de capter le champs magnétique induit par ce courant. Bien sûr, ce système est sensible à la transmission entre l'émetteur et la bobine.

Plusieurs systèmes de conduite automatique ont été réalisés à la fin des années 60 et au début des années 70, mais aucune réalisation n'a été retenue. Par exemple, un système de contrôle électronique d'un bus réalisé par Barrett Electronics en 1961 [Tes61], et des véhicules évoluant à trés basse vitesse dans des hôpitaux et dans des entrepôts [Ric66] testés vers 1966 n'ont pas duré longtemps.

Dans le domaine de l'automatisation des systèmes de transports ("guideway et busway"), plusieurs recherches réalisées entre 1960 et 1970, ont été implémentées et sont opérationnelles aujourd'hui dans le service public et certaines d'entre elles avec d'excellentes performances.

Développement de la théorie sur le contrôle latérale

A l'exception des "Cabinentaxi" [Bar62], les développements des contrôleurs latéraux dans les années 60 et au début des années 70 n'ont pas de fortes bases théoriques, mais ils sont plutôt basés sur des approches empiriques. *Pasternack* est le premier a avoir pris conscience de cette faiblesse. Il a utilisé les principes de l'automatique moderne pour établir une commande avec un modèle dynamique à deux degrés de liberté en utilisant la mesure de l'erreur entre la ligne à suivre et un capteur monté au centre de gravité du véhicule [Pas71, Pas73]. Parallèlement à ce travail, *Bonderson* de GM [Bon74] et *Shladover* du MIT [Shl76], ont appliqué la théorie de la commande optimale pour réaliser des contrôleurs latéraux optimaux. *Bonderson* a utilisé la théorie de la commande linéaire temporelle (résolution de l'équation de Riccati), par contre *Shladover* a adopté la théorie du filtre de *Wiener* dans le domaine fréquentiel pour résoudre le problème du contrôle latéral. Les deux chercheurs ont obtenu des résultats comparables.

Les recherches de MIT ont continué avec l'utilisation de la théorie de la commande optimale pour faire le suivi de ligne [SWRF78], avec traitement des effets transitoires associés aux courbures [SFWR77]. Cette théorie a été adoptée par *Panik* et ses collègues de Daimler Benz [CDPW], avec une importante contribution de vérification expérimentale sur un bus utilisant un fil électrique comme référence de guidage [Dar87, Wei88]. Ce travail est l'un des plus important jamais réalisé dans le domaine du contrôle latéral. En effet, il a été étendu à des tests avec des passagers à Fürth (Allemagne) utilisant une référence électronique, et par la suite utilisé quotidiennement dans des opérations de transit à Essen (Allemagne) et à Adelaide (Australie) avec une référence mécanique. Le système de référence électronique a été récemment mise en œuvre pour le contrôle latéral des véhicules spéciaux opérant dans Eurotunnel.

Utilisation de la vision

Tous les travaux décrits jusqu'ici utilisent des références spécifiques installées sur ou sous les routes. Des chercheurs de la communauté de l'intelligence artificielle ont pensé utiliser les techniques de vision pour déterminer l'orientation du véhicule relativement aux lignes de marquage conventionnelles. Leurs efforts étaient concentrés sur le traitement d'image plutôt que sur les problèmes liés à la dynamique et à la commande des véhicules. Ces travaux sont issus de l'industrie militaire qui s'intéresse au guidage de véhicules autonomes tout terrain. L'application de cette technologie sur des véhicules terrestres peut être trouvée dans le papier de *Metzler* de Daimler Benz [Met88].

Le travail pionnier dans le domaine du guidage par vision a été fait par le *Pr. Ernst Dickmanns* et ses collègues de l'UBM (**U**niversitat der **B**undeswehr de **M**unich) [DZ86, DZ87, Gra93]. Les articles décrivant le guidage latéral par vision sont très nombreux, mais la plus part d'eux ne s'intéresse pas aux problèmes de commande. Pour un état de l'art de l'application de la vision, on peut citer respectivement le livre [Mas92] et les conférences organisées par *Masaki* intitulées IEEE Intelligent Vehicles Symposium depuis 1992.

Le contrôle latéral dans le programme PATH

En 1988, les recherches sur le contrôle automatique latéral ont commencé à l'UCB (**U**niver- sity of **C**alifornia **B**erkeley) dans le cadre du programme PATH . Le système de commande des véhicules est basé sur un capteur qui détecte la position des véhicules relativement à des aimants permanents introduits dans les routes. Les travaux ont commencé par le développement des modèles dynamiques de véhicule et des lois de commandes, leurs évaluations par simulation avant de commencer les expérimentations. L'analyse de ces travaux est décrite par *Peng et Tomizuka* dans [PT90, PT91]. La théorie du contrôle optimal est utilisée pour maximiser le confort. *Peng et al.* [PZS+93, PZS94] dirigent les expérimentations en collaboration avec IMRA America pour démontrer la performance de ce contrôleur et de le comparer avec un simple PID. Les expérimentations ont montré que le suivi à une bonne précision avec des courbes assez fortes ($0.27\ g$ en accélération latérale) et sous des conditions anormales (variation du poids, pneu dégonflé, surface sous la neige). L'erreur latérale est généralement inférieure à $10\ cm$, et moins que $15\ cm$ dans les plus mauvais cas. Les performances ont été évaluées sur un véhicule et par plusieurs conducteurs [PZS+93], et le système de contrôle qui, initialement était réalisé pour une Toyota Celica a été re-implémenté sur une Pontiac 2000 avec un minimum de modifications, tout en gardant d'excellentes performances [PZS94].

Les travaux du programme PATH ont été étendus pour réaliser des tests d'un contrôleur alternatif, utilisant la logique floue [HT94]. Le contrôleur sera destiné aux manœuvres de changement de ligne [CT94]. Une commande optimale qui tient compte des glissements a été réalisée, mais elle n'est pas encore validé expérimentalement. Deux autres contrôleurs latéraux basés sur la détection des mouvements des véhicules de tête plutôt que de se référencer aux plots magnétiques (dans le but de minimiser le coût dans certaines routes) sont en cours d'études. *Narendren* et *Hedrick* ont proposé un suivi en train de véhicules [NH94] avec un contrôle des glissements. *Fujioka* a proposé un contrôleur de suivi de trajectoire, utilisant aussi un contrôleur de glissement, où le véhicule de tête doit communiquer sa trajectoire aux véhicules suiveurs [FS94]. Aucune de ces approches n'est vérifiée expérimentalement.

Plusieurs recherches ont été entamées pour augmenter la robustesse des contrôleur latéraux. *Ackermann* [AS90, GUA94, ASSU94] a utilisé des approches analytiques pour trouver des contrôleurs à gains fixes qui peuvent être performants sur une grande gamme de vitesse, avec une variation du poids et sur plusieurs états de route. Des simulations, sans informations sur l'état de la route, ont montré de bonne performances. [GUA94, ASSU94]. *Patwardhan* et *Tomizuka* [PTZD94] ont réalisé un système de contrôle qui peut être perturbé par les changements des dynamiques de véhicule, par l'explosion des pneus. Le deux chercheurs ont fait des tests en boucle ouverte sur un véhicule réel avec dégonflement des pneus, mais n'ont pas testé le contrôleur en boucle fermée. *Smith* et *Starkey* [SS94] s'interessent au contrôle latéral automatique pour les manœuvres d'urgences où l'accélération latérale peut être importante.

Les références et les capteurs latéraux

La majorité des recherches dans le contrôle latéral sont basées sur deux catégories de références : soit un fil électrique actif au milieu d'une ligne ou bien des lignes peintes. Cependant, d'autres systèmes ont été étudiés. *Mayhan* et *Bishel* [MB82] ont expérimenté des radars qui pouvaient recevoir des signaux grâce à un grillage installé le long de la route. La précision de ce capteur est environ deux fois plus grande que celle du système du fil électrique utilisé par l'OSU. Récemment, le radar a été repris par *Giubbolini* et *Cugiani* en Italie [GC93, CG94] et par *Johnson et al.* [Joh93] aux USA pour détecter des matériaux posés sur la surface de la route. *Bishel* a tenté d'utiliser le système radar-grillage comme référence pour faire des tests de suivi [Bis94]. *Johnson et al.* [JAL79] ont utilisé des marques discrètes comme références de guidage à très basse vitesse. *Yasui* et *Margolis* [YM92] décrivent la conception d'un système qui utilise des réflecteurs comme lignes de référence et un capteur optique qui, par ballayage, permet de donner les informations de l'orientation du véhicule. *Zhang* et *Parson* [ZP90] ont proposé l'utilisation d'aimants permanents pour marquer la référence du contrôle latéral. Une série d'expériences a été réalisée démontrant, ainsi que leur capteur pouvait avoir des informations extrêmement précises ($5\ mm$). Ces marqueurs magnétiques, qui peuvent être installés suivant un code binaire (pole N ou S) pour coder la courbure du chemin, ont servi comme référence dans les expérimentations du programme PATH.

Résumé des connaissances sur le contrôle latéral

Basé sur les recherches entamées depuis longtemps, l'état actuel sur la connaissance du contrôle automatique latéral apparaît comme suit :

- Plusieurs couple de références/capteurs paraîssent techniquement réalisables grâce aux démonstrations effectuées sur des prototypes. Ces systèmes se distinguent les uns des autres par leur précision, par leur sensibilité vis à vis des perturbations de l'environnement, et surtout par le coût imposé lors des installations routières. Finalement, le coût économique doit être un facteur déterminant dans le choix.

- Les systèmes de contrôle latéral automatique ont démontré une capacité de précision du suivi nettement supérieure aux capacités des conducteurs humains.

- Il est important que les systèmes de contrôle latéral soient capables de traiter les échecs.

- Il apparaît possible d'améliorer les performances des contrôleurs latéraux vis à vis des conditions anormales, comme la variation du poids du véhicule, le type de contact sol-pneu, roues anormales, etc..., mais ceci n'est pas encore prouvé expérimentalement.

- Un système robuste sur une grande gamme de forces extérieures n'a pas été encore prouvé. Ces forces sont constituées non seulement des forces du vent, mais aussi des forces aérodynamiques des différents types de véhicules; les forces de contacts pneus-sol et même les collisions modérées avec d'autres véhicules.

– Des bonnes actionneurs latéraux sont nécessaires pour donner une bonne performance au système de contrôle latéral.

1.3.3 Etat de l'art de la recherches sur le contrôle longitudinal

Les méthodes du contrôle longitudinal sont appliquées depuis longtemps sur les chemins de fer avec un guidage mécanique. Alors que le contrôle latéral est une fonction individuelle pour un véhicule qui est guidé relativement à une référence, le contrôle longitudinal est le résultat d'une interaction du contrôle de plusieurs véhicules. La diversité des approches dans le contrôle longitudinal est établie de plusieurs manières suivant la prise en compte de ces interactions ou leur évitements. Il existe trois approches de contrôle longitudinal de véhicule :

– contrôle de bloc fixe;

– suivi de point (ou contrôle de bloc mobile);

– suivi de véhicule

Le contrôle de bloc fixe, qui est utilisé en standard les systèmes de chemins de fer automatiques, est la plus ancienne et la plus simple des approches. Cependant c'est elle qui limite le plus les capacités de mobilité du véhicule. Dans cette forme de contrôle, le chemin est divisé en segments, appelés blocs, et les véhicules ne peuvent entrer dans un bloc déjà occupé par un autre véhicule.

Les systèmes de suivi de point consiste à se déplacer en se référant à une référence virtuelle le long de la route. Les véhicules sont commandés pour la suivre avec le plus de précision, ce qui leur donne une plus grande capacité que le systèmes de blocs fixes, mais complique plus le contrôle. Le suivi de point nécessite d'une part l'installation d'équipements sur les bords de la route afin de générer la référence, et d'autre part une centralisation des fonctions d'administration du système (planification des mouvements du véhicule, avec synchronisation ou quasi-synchronisation des plans).

Les deux systèmes, de contrôle de bloc fixe et de suivi de point fournissent une base de contrôle des opérations du véhicule dans des chemins non bouchés. Cependant, dans ce mode, les véhicules ne voient pas ce qui se passe à l'avant à moins qu'ils soient prévenus avec des systèmes de détection d'obstacle. De pareils systèmes ont probablement besoin de détecter tout objet pouvant intervenir dans leur chemin et pouvant causer des collisions.

Les systèmes de suivi fournissent aux véhicules tout l'essentiel des fonctionnalités de contrôle au niveau de régulation, indépendamment des bords de routes. Ceci permet de transférer le coût des installations routières vers les véhicules, en augmentant considérablement la complexité du système embarqué, mais en lui permettant une plus grande autonomie qu'avec les systèmes de suivi de point. Les véhicules doivent utiliser des capteurs pour constituer leur état relativement à leur voisins. Ces mêmes capteurs peuvent aider à la détection des obstacles, et des autres véhicules pour éviter les collisions. Les systèmes de suivi de véhicules ont l'avantage d'être compatibles avec une administration asynchrone et une philosophie de contrôle complètement distribuée (par opposition au système centralisé).

Les systèmes de contrôle longitudinal des blocs-fixes

Ces systèmes n'utilisent qu'une information binaires (si le bloc est occupé ou non par un véhicule). Ceci introduit une grande erreur de quantification sur la localisation, et rend difficile la détermination des distances entre les véhicules lors de l'apparition inattendue d'un véhicule. Toutefois, l'approche a été appliquée sur les systèmes AGT (Automated Guideway Transit) [RHM73, SHR72] avant que les défauts ne soient identifiés.

Des améliorations peuvent être effectuées en associant à l'information sur la position, la vitesse du véhicule. Cette idée a été mise en évidence par le concept "Augmented Block Guidance" dans le laboratoire APL (**A**pp- ied **P**hysics **L**aboratory) de l'université de Johns Hopkins [Pit72, PH73, Pit71]. Les chercheurs de ce laboratoire considèrent un système avec six blocs, où un véhicule se trouvant à six blocs derrière un autre évolue librement. Par contre, des contraintes se poseront sur lui à l'approche de son prédécesseur. Les mesures de la rotation des roues doivent permettre d'une part d'estimer précisément la position du véhicule à l'intérieur de ces blocs, et d'autre part de déterminer la vitesse du véhicule afin d'augmenter la réponse du système. L'évitement d'obstacle est proposé par l'utilisation de l'approche "plan de phase" pour décider (en fonction de la vitesse et l'inter-distance) quand le freinage d'urgence sera activé.

Whitney et *Tomizuka* [WT72] ont réalisé un contrôleur PID qui rajoute, à l'information sur la position du bloc, une boucle en vitesse pour enlever les erreurs statiques de position. Ils ont démontré que ce contrôleur est équivalent à celui de *Wilkie* (décrit ci-dessous), qui est efficace dans les conditions normales. Cependant, dans les situations d'urgence, il introduit des oscillations en augmentant les distances d'arrêt. Il doit avoir besoin d'un retour en accélération pour améliorer ses performances.

Systèmes de contrôle longitudinal de suivi de point

Les systèmes de contrôle longitudinal de suivi de point ont intéressé considérablement les chercheurs des systèmes AGT (**A**utomated **G**uideway **T**ransit) et PRT (**P**ersonal **R**apid **T**ransit) dans les années 60 et 70, et quelques uns d'entre eux ont été implémentés dans des systèmes opérationnels.

On peut citer, deux tentatives de conception des contrôleurs de suivi de point, n'utilisant que l'erreur sur la position longitudinale (distance entre le véhicule et le point de référence désiré). La première, par TRW, négligeant les dynamiques longitudinales du véhicule [trw69]. La deuxième, par *Smith* [Smi], utilisant un contrôleur optimal et l'observateur de *Luenberger* pour estimer le reste de l'état du véhicule à partir de l'erreur en position longitudinale.

Plusieurs recherches tiennent compte de le vitesse du véhicule en plus de l'erreur en position dans la boucle d'asservissement. *Hinman* [Hin75] a proposé une commande en vitesse avec un régulateur PI sur l'erreur en position pour avoir une bonne réponse dynamique. Dans le système CVS, au Japon, *Ishii et al.* [Ish74] ont appliqué un régulateur PI sur l'erreur en position pour trouver une estimation de l'erreur en vitesse.

Fenton et *Chu* [FC77] ont développé un contrôleur de suivi de point en utilisant comme variables de mesures, l'inter-distance entre le véhicule et le point de référence, et la différence entre la vitesse du véhicule et la vitesse du point de référence. Après, ils ont raffiné

leur contrôleur [FM91] avec un PI sur la vitesse en tenant compte de la non linéairité de celle ci à cause des glissements des roues et des dynamiques du système de propulsion. Par simulation, ils ont obtenu $42\ cm$ d'erreur maximale en position dans les conditons normales, et $1.2\ m$ avec un freinage de $0.45\ g$.

En se basant sur l'hypothèse de la disponibilité complète et continue des informations sur l'état du véhicule relativement à son point de suivi, des recherches ont été réalisées en utilisant la théorie de la régulation optimale. Un article utilisant cette théorie est décrit par *Wilkie* [Wil70]. *Fling* et *Olson* [FO72] ont étendu les travaux de *Wilkie* en tenant compte de l'échantillonnage des données des marqueurs sur la route. D'autre extensions ont été proposées par *Garrard* et *Kornhauser* [GK73a, GK73b] avec l'utilisation d'un modèle complexe pour traiter le confort des passagers et des observateurs pour estimer toutes les variables de l'état du système à partir des mesures des erreurs de la position et de la vitesse. Travaillant séparément, *Kornhauser* [Kor74] a appliqué le filtre de *Kalman* pour tenir compte des effets du bruit d'échantillonnage, et *Garrard* [YG74] a traité la robustesse du contrôleur vis à vis des forces extérieures et des variations de la masse du véhicule.

Une seule équipe de recherche, à Alden "Self Transit Systems Corporation", a traité le problème du suivi de point en tenant compte des effets du freinage d'urgence [EW74, WW74, Whi79]. Dans ces études, la décision du freinage est binaire (mettre le frein ou non), suivant l'erreur en position et relativement à un seuil [EW74]. Par suite, des améliorations ont été faites en exprimant le seuil comme une fonction de la vitesse [WW74] ou de l'accélération [Whi79].

Les publications sur le contrôle longitudinale de suivi de point ont disparu à partir de 1980, dés que la technologie a commencé à permettre la réalisation d'un système plus intelligent pouvant faire le suivi de véhicules.

1.3.4 Systèmes de contrôle longitudinal de suivi de véhicule

Les travaux de recherche sur le contrôle longitudinal des systèmes de suivi de véhicules sont très récents. Ces systèmes sont plus compliqués que les deux premiers et offrent plus de choix de conception, et surtout ils peuvent augmenter considérablement le débit sur des routes. Notons que peu de réalisation sont autonomes : c'est à dire qu'elles fonctionnent sans communication entre les véhicules.

Les publications les plus récentes, sont basées sur les approches de l'automatique non linéaire et invariante dans le temps. Très peu de structures de contrôle non linéaires et variantes dans le temps ont été implémentées et vérifiées expérimentalement sur des véhicules réels.

Le choix de la distance de suivi reste un aspect trés discutable. Trois classes du suivi de véhicules sont envisageables :

- inter-distance constante,
- temps constant,
- facteur de sécurité constant

Les systèmes avec une inter-distance constante sont les plus appropriés à l'intérieur des trains de véhicules. Quand ils sont bien réalisés, ces systèmes donnent une illusion d'un couplage mécanique rigide entre les véhicules, cependant ils nécessitent plus d'informations que les autres classes pour avoir une conduite asymptotiquement stable. Les systèmes avec un temps constant peuvent être réalisés avec une stabilité asymptotique en utilisant seulement les informations reçues à partir du véhicule suivi, mais l'espace varie proportionnellement avec sa vitesse pour maintenir le temps constant. Les systèmes avec un facteur de sécurité constant peuvent être utilisés avec plusieurs critères de sécurité, mais typiquement ils maintiennent la distance proportionnelle à v^2.

Asservissement par bouclage sur la distance et la vitesse du véhicule précédent La structure de contrôle la plus simple est celle étudiée par *Brown* [Bro74], utilisant seulement la mesure de la distance du véhicule précédent et la vitesse du véhicule suiveur. Les études de *Brown* ont montré une performance satisfaisante avec un PI sur l'erreur en distance, combiné avec un bouclage proportionnel à la vitesse du suiveur. Cette approche n'a jamais été testée sur des véhicules en présence de perturbations et de bruits.

Asservissement par bouclage de la distance relative et de la vitesse relative avec le véhicule précédent C'est un contrôleur très pratique par sa simplicité et sa possibilité d'être utilisé dans le trafic mixte, où seulement un ou quelques véhicules sont équipés pour faire le suivi. La différence de vitesse est directement mesurée par un capteur, ou bien elle est obtenue par communication de la différence des vitesses entre le véhicule contrôlé et le véhicule précédent.

Les premiers rapports décrivant la structure de ce contrôleur sont apparus dans le groupe de recherche dirigé par *Fenton* à l'université de l'OSU [BF70, Fen68, BFO71]. Les chercheurs de l'OSU ont divisé les différentes phases de vitesse en six régions et ont défini des lois de commande différentes pour chaque région. Après simulations, ce contrôleur a été testé sur deux véhicules dans une piste d'essayage [FM91]. Ce groupe de recherche a basculé ces recherches de suivi de véhicule vers le suivi de point en 1971 parce qu'il n'a pas trouver le capteur adéquat, ayant une précision de $\pm 0.3\ m$ sur une distance comprise entre $10\ m$ et $100\ m$ [FM91].

Caudill et *Garrard* ont étudié la même structure pour réaliser des simulations [Gar77, CG77a, CG77b]. Dans leurs travaux, les bouclages de l'inter-distance et de la vitesse relative ont été augmentés par l'anticipation sur la vitesse du véhicule précédent. Un PID a été utilisé pour la régulation de l'inter-distance et un PI pour la régulation de la vitesse. Ils ont évoqué l'effet de l'accélération dans le contrôle du système, et ont montré comment elle peut provoquer une résonnance à moins que les limites de la saturation ne soient atténuées en changeant les gains (gains non linéaires). Par la suite, *Garrard* a repris ce travail en considérant les effets de la limite du jerk sur la stabilité [Gar78]. *Olson* et *Garrard* sont allés encore plus loin en développant deux contrôleurs de suivi longitudinal, l'un d'eux utilise l'inter-distance et la vitesse relative comme entrée, et l'autre ajoute encore l'accélération relative.

Rouse et *Hoberock* ont utilisé les mêmes variables de bouclage pour analyser le suivi de véhicule dans un train, mais ils n'ont retenu qu'un modèle dynamique simple de troi-

sième degré, avec un degré pour représenter la dynamique du système de propulsion [RH76, HR76]. Ils ont choisi la technique de suivi avec "facteur de sécurité constant" à l'intérieur de leur train.

Plus récemment, cette structure de bouclage est devenue très populaire dans les systèmes AICC (Autonomous Intelligent Cruise Control). Les innovations technologiques des capteurs ont encouragé les industriels de l'automobile à réaliser des "voitures suiveuses" pour une éventuelle industrialisation. Les industriels projettent de faire de l'assistance au conducteur pour augmenter le confort, et ne pensent pas encore à automatiser les fonctions de conduite.

Zhang [Zha91] a défini plusieurs niveaux de contrôle longitudinal dans PROMETHEUS allant des systèmes de prévention, AICC, à la réalisation d'un petit train avec communication entre les véhicules. Son papier définit les concepts mais ne donne pas des résultats. *Benz* et *Zhang* [ZB93] ont défini un modèle pour évaluer les performances et la sécurité offerte par les systèmes AICC, sans montrer les caractéristiques d'un système spécifique. Les expériences de PROMETHEUS avec ces systèmes sont résumées par *Becker et al.* dans [BBD$^+$94a] et [BBD$^+$94b]. Ces deux articles décrivent les problèmes de l'interface humaine plutôt que les dynamiques du véhicule et la conception des lois de commande. Les sujets techniques sont détaillés plus spécifiquement par *Sala* et *Pressi* dans [SP94], où un parcours des différents critères de suivi utilisés dans les travaux de PROMETHEUS en AICC est retracé. Cet article contient ce que nous sommes en train d'expliquer, plus spécifiquement sur l'estimation de la distance de sécurité. Il décrit aussi, des stratégies de coopération avec communication entre les véhicules.

Les travaux au Japon et aux USA, dans le domaine de l'AICC sont un peu en retard sur l'Europe. *Satoh* et *Taniguchi* [ST94a] décrivent un système basé sur l'utilisation d'un radar laser et appliquent la commande en essayant de suivre le comportement d'une conduite normale. Leurs expériences sur un prototype ont montré que les conduites sont très confortables, mais le papier ne donne aucune indication sur les effets de ce système sur le débit du trafic, ni sur la stabilité de ce contrôleur quand il équipe plusieurs véhicules. *Ioannou* et ses étudiants ont appliqué la structure de ce contrôleur pour automatiser des véhicules dans des routes réservées [CI92, CIL94]. Ils ont montré que leur contrôleur pourrait avoir une stabilité asymptotique avec une simple boucle d'asservissement basée sur le critère de temps constant. Les simulations et la conception du contrôleur se sont suivies avec des implémentations sur des prototypes [IX94].

Une grande quantité de rapports techniques sur l'AICC a été produite par *Paul Fancher* et ses collègues à l'Université de Michigan, et en collaboration avec "Leica" qui leur a développé l'électronique embarquée pour la réalisation des expérimentations [FBJ94, FB94a, FB94b, FEB$^+$94]. Le contrôleur réalisé est un PI avec saturation de l'action intégrale [FBJ94]. Leur véhicule était équipé d'un capteur infra-rouge permettant de déduire directement l'information sur la distance. Ils ont conçu soigneusement un contrôleur d'opération en définissant un plan de phase à partir de l'information sur la distance [FB94a]. Et, ils ont évalué les effets probables de l'AICC sur le trafic par des simulations des manœuvres de décélération des véhicules automatiques. Ils ont montré que leur système est asymptoti-

quement stable si l'intervalle de temps, entre les deux véhicules, est deux fois supérieur à la constante de temps du système de contrôle [FB94b].

Asservissement par bouclage de la distance et de la vitesse du véhicule suiveur plus un bouclage sur la vitesse par rapport au sol La stabilité asymptotique de la conduite du véhicule peut être prouvée en ajoutant à tous les véhicules, un asservissement par rapport à la vitesse du sol. Dans cette structure de contrôle, la commande en vitesse est par rapport au bord de la route. *Foster* [Fos79] a proposé une originalité, en utilisant un bouclage sur la vitesse absolue pour prouver la stabilité asymptotique des systèmes utilisant la technique de l'inter-distance constante. D'autres structures de contrôle, que nous traiterons par la suite, peuvent rendre ces systèmes plus stables par l'ajout de cette boucle.

Palmquist [Pal93] propose une structure de contrôle où un centre de contrôle du trafic régule une vitesse commune à tous les véhicules équipés avec ce système. Le but est d'harmoniser le trafic en évittant les dépassements de vitesses pour mettre les véhicules en sécurité. Ce papier décrit techniquement l'approche et traite son implémentation. Les expérimentations ont été concluantes en Suède en utilisant la communication pour établir la commande en vitesse par rapport à la route.

Asservissement par bouclage sur la distance et de la vitesse du véhicule suiveur plus une commande en accélération externe *Chiu*, *Stupp* et *Brown* du laboratoire APL (**A**pplied **P**hysics **L**aboratory) de l'université de Johns Hopkins ont étudié ce contrôleur en publiant plusieurs articles à la fin des années 70 [CSB76, CSB77, Chi79]. Ils ont défini une frontière cinématique pour limiter les rapprochements des véhicules avec leurs prédécesseurs en se basant sur les mouvements anormaux de ces derniers. Ils ont aussi attiré l'attention sur les limites d'un contrôleur linéaire invariant dans le temps sur les courtes distances. En effet, si les gains sont choisis suffisamment grands pour un contrôle serré (sur des courtes distances), ceci risque de produire des actions de contrôle excessives en grandes distances. Ce problème est traité en donnant des gains qui dépendent du temps durant les manœuvres de dépassement en tenant compte des saturations de l'accélération et du jerk. C'est aussi, les premieres recherches à avoir reconnu les effets de la déstabilisation causée par les limitations du jerk.

Cette structure a été appliquée très récemment pour le contrôle de suivi de véhicule en Allemagne et au Japon. *Müler* et *Nöcker* de Daimler Benz décrivent un contrôleur AICC utilisant un capteur intra-rouge et utilisant la logique floue pour essayer de faire une expérience de conduite [MN92]. L'implémentation a été réalisée sur un prototype, et son article présente les résultats de ses tests et indique que les règles floues ne sont pas encore parfaites. Cette structure a aussi été appliquée par *Fujita et al.* de Honda [FAS94] pour un système de freinage d'évitement d'obstacles. Le système est implémenté sur un prototype, et a été évalué à travers plusieurs tests.

Asservissement par bouclage de l'état "complet" du véhicule précédent et du véhicule contrôlé (distance, vitesse et accélération relatives) Apparemment, cette structure a

été décrite dans plusieurs articles, sans qu'elle soit testée. *Olson* et *Garrard* l'ont comparée avec d'autres structures de modèle de contrôle sans bouclage d'accélération dans [OG79].

Asservissement par bouclage de l'état "complet" du véhicule précédent plus la vitesse commandée du véhicule précédent Cette structure a été proposée par *Tsugawa* [YHTM89, TM90] dans leur concept "soft linked". Ce concept a été proposé pour être implémenté en utilisant des communications entre les véhicules plutôt que d'utiliser un capteur de distance. Chaque véhicule a besoin de savoir sa position absolue pour éviter de mesurer l'inter-distance. Au début, la proposition était faite pour faire de la commande à petites vitesses à l'intérieur des usines ou les entrepôts, avec une communication infrarouge entre les véhicules successifs. Après elle a été étendue, avec des simulations, sur les véhicules routièrs, sans être implémentée expérimentalement.

Asservissement par bouclage de l'état "complet" du véhicule précédent et la vitesse commandée plus le jerk du véhicule précédent Cette structure a été analysée en profondeur par *Pue* du laboratoire APL à la fin des années 70 [Pue77a, Pue77b, Pue78, Pue79] pour surmonter les inconvénients des variations temporelles des gains dans les travaux de *Chiu*, *Stupp* et *Brown*. *Pue* a développé une famille de contrôleurs non-linéaires assez compliqués. Ses contrôleurs optimums et sous-optimums utilisés avec la structure de contrôle citée dans avant ont été validés par simulation sur un modèle à trois intégrations sans la dynamique de propulsion. Le contrôleur est asymptotiquement stable et a de bonnes performances. Et puisque cette approche nécessite une communication des données (car elles ne peuvent pas être mesurées directement), ces études tenaient compte du débit des données et des effets des imperfections de la communication (échantillonage, quantification et délai) sur la performance de la commande [Pue79].

Asservissement par bouclage sur la distance et la vitesse relative du véhicule précédent plus la vitesse relative du véhicule de tête Cette structure est proposée par *Shladover* [Shl78] pour prouver la stabilité asymptotique du système de suivi de véhicule avec une inter-distance constante. Les informations venant du véhicule de tête (les données doivent être disponibles immédiatement par communication plutôt qu'avec des capteurs) servent comme référence absolue pour atténuer les perturbations. Cette approche permet d'une part d'éviter de déstabiliser le suivi par les saturations du jerk et de l'accélération et d'autre part d'avoir un asservissement performant qui peut maintenir très bien un espace constant entre les véhicules.

Asservissement par bouclage de l'état "complet" du véhicule contrôlé et précédent, plus la vitesse et l'accélération du véhicule de tête Cette extension de la structure précédente a été développée par les chercheurs du programme PATH, pour donner plus de flexibilité dans les dynamiques dans la boucle d'asservissement des divers trains du véhicules avec une inter-distance constante entre les trains. L'addition de la décélération dans la boucle rend possible de placer les pôles et les zéros dans les fonctions de transferts en boucle fermée, sans ajouter de communications supplémentaires. Ceci a été appliquée sur

un modèle relativement simple, de troisième ordre avec quelques effets de non-linéarité [Shl91, SD90] et sur un modèle qui tient compte des dynamiques de la chaine mécanique de transmission du moteur [MHS90, HNS91, HS94]. Les approches de la conception du contrôleur sont le placement de pôles [Shl91, SD90] et le contrôle non-linéaire des glissements [MHS90, HNS91, HS94].

Ces analyses et simulations ont démontré une stabilité asymptotique des véhicules dans le train dans les conditions idéales, mais aussi en présence des imperfections dans l'environnement comme le bruit de mesure, et les erreurs de quantification et d'échantillonnage [Shl91]. Une étude comparative [SD91] a illustré une diminution de la stabilité asymptotique suite à la perte de communication avec le véhicule de tête. Une autre étude [SD92b] montre les effets des délais de communications, de la différence de performances entre les véhicules, et de l'influence des bruits de mesure.

Une famille de contrôleurs, basée sur la conception de contrôle de glissement a été évaluée, par *Hedrick* et ces collègues [MHS90, HNS91, HS94], par de nombreuses expériences, avec deux véhicules et puis sur un train de quatre véhicules. Les premières expériences n'ont pas été concluantes sur la stabilité asymptotique à cause des imperfections du capteur de position [CLD$^+$91]. Plus tard en améliorant le capteur de position, les expérimentations ont donné de meilleurs résultats avec une excellente précision et une très bonne qualité de confort [CH95a, CH95b]. Même avec un train de deux véhicules, les premières expériences ont montré l'importance d'un actionneur sur l'accélération très rapide pour atteindre un suivi stable. Les expériences avec quatre véhicules ont démontré une capacité d'opération avec une inter-distance de $4\ m$ avec $30\ cm$ de variation, une accélération de $0.1\ g$ et une décélération de $0.05\ g$ sur une pente maximale de 3%. Ces expérimentations ont été faites en commandant l'accélérateur mais pas les freins. En effet des études sur l'actionneur de freinage ont démontré une inadaptation de la réponse dynamique du système de freinage qui agit sur la pédale de freinage et non directement sur le circuit de freinage [MGH94].

Asservissement par bouclage de l'état "complet" du véhicule contrôlé et les N véhicules suiveurs Cette structure a été appelée "Continuous Paltooning" par *Ren* et *Green*. Des analyses dans [RG94] indiquent ses avantages dans la réponse vis à vis des perturbations du véhicule de tête sur les véhicules contrôlés. La charge de la communication dépend du nombre de véhicules dans le train. Cette approche a été analysée plutôt par *Athans* [ALL66].

Asservissement par bouclage de l'état "complet" entre tous les véhicules Un contrôleur optimal a été premièrement proposé par *Athans* dans le cas continu [ALL66] puis dans le cas discret [LA68]. La théorie de la commande optimale a été utilisée pour choisir les gains, et des simulations ont montré une bonne performance dans les conditions normales. Basé sur la technologie des années 60, ce contrôleur suppose que les mesures de chaque véhicule doivent être communiquées vers un ordinateur de contrôle. *Powner et al.* [PAQ69] ont traité le même problème en utilisant la programmation dynamique pour essayer d'avoir des implémentations temps réel efficace. *Peppard* et *Gourishankar* [PG72] ont réduit un

petit peu les dimensions de ce problème en éliminant de la loi de commande la boucle des chemins.

Autres contrôleurs Il existe une autre classe de contrôleurs qui n'ont pas un bouclage bien définit comme dans les structures décrites avant. Ils incorporent des modèles plus complexes que les exigences du suivi. Depuis la formulation du modèle de suivi par *Olson* et *Garrard* [OG79], d'autres formes ont fait leur apparaition. Par exemple, *Sklar et al.* [SBS79] définissent un système de modèle-suivi basé sur la construction d'un modèle complexe. Un simple régulateur en vitesse pour forcer le véhicule à suivre la trajectoire du modèle est réalisé. Ce modèle incorpore les contraintes cinématiques basées sur la distance de sécurité. *Caudill* et ses étudiants [CB80] [CMT82] ont aussi étudié un modèle de suivi pour les véhicules routiers, et les poids lourds, en insistant sur les vitesses de contrôle. *Youcef-Toumi et al.* [YTSAH92] décrivent aussi plusieurs modèles de système de suivi qui utilisent seulement la mesure de la distance pour implémenter la loi de commande. Ceci a été testé en laboratoire avec un modèle réduit à une échelle de $1/5$.

Aucun modèle de système de suivi n'a été implémenté sur un prototype réel, mais les simulations réalisées dans des conditions idéales ont donné de bonnes performances.

La conception du contrôle pour le suivi de véhicule est toujours basée sur l'hypothèse que le véhicule de tête suivi est bien connu. Cependant, dans le cas où un autre véhicule se met dans le champ du véhicule contrôlé, ce dernier doit lancer la loi de commande appropriées pour le suivi. Ce problème a été initialement traité par *Rouse* et *Hoberock* [RH76, HR76] en supposant l'utilisation de la technique, mentionnée dans la section 1.3.4, de contrôle de bloc. Plus récemment, *Goodbole* et *Lygeros* [GL94b] ont proposé une loi de commande du véhicule de tête avec plusieurs modes d'opérations.

Capteurs pour le contrôle longitudinal

La littérature sur les capteurs est très vaste et généralement elle est différente de la littérature décrivant la dynamique et le contrôle des véhicules. Généralement, les capteurs de positionnement sont des radars. *Grimes* [GG89] donne une description très détaillée de son radar embarqué. Le projet SMILER (**S**hort range **MI**crowave **L**ink for **E**uropéen **R**oads) s'interesse à la réalisation de radars [ADL$^+$91]. Plus récemment, *Philips* a conçu un radar dont une présentation pas très technique peut être trouvée dans [AW93]. *Kikuchi et al.* donnent dans [KIN94] des informations sur les développements de Honda d'un radar laser.

Résumé des connaissances sur le contrôle longitudinal

La recherche sur le contrôle longitudinal est très riche de par les divers approches validés par simulation. Cependant, peu d'approches ont été testées expérimentalement là où les imperfections des actionneurs et des capteurs dans l'environnement, les variations des paramètres, le bruit et les forces extérieurs existent. Les axes de recherches à retenir sont les suivants :

– Les développements dans les domaines de l'informatique, des technologies des cap-

teurs, de la communication et dans le contrôle, ont portés la recherche vers les systèmes de suivi de véhicules en délaissant les systèmes de suivi de bloc et de suivi de point. Bien que ces systèmes nécessitent une technologie plus compliquée, ils ont l'avantage d'augmenter le débit des véhicules sur les routes, d'être autonome et tolérants aux fautes, et d'un coût total moindre.

– Durant des années, une très large variété de structures de contrôle a été proposée et évaluée, cependant seulement quelques une d'entre elles ont été vérifiées expérimentalement. Des structures peuvent être implémentées dans des véhicules autonomes (en utilisant que des capteurs embarqués), mais la plupart nécessitent une communication entre les véhicules.

– Les systèmes de suivi de véhicules doivent être asymptotiquement stables pour un nombre donné de véhicules. La stabilité doit être assurée dans des conditions d'opérations réalistes, en tenant compte des délais des communications, du bruit des capteurs, des erreurs de quantification et d'échantillonnage, des variations de paramètres...

– Plusieurs critères peuvent être appliqués pour déterminer la relation entre l'inter-distance et la vitesse. Elles peuvent être : une distance constante, un temps constant ou un facteur de sécurité constant. La communication n'est pas toujours nécessaire pour un contrôle stable. Par exemple, *Swaroop et al.* [SHCI94] a réalisé un contrôleur longitudinal asymptotiquement stable, grace à un capteur autonome, en gardant un temps constant entre les véhicules. Par contre pour réaliser un suivi asymptotiquement stable avec une distance constante, il faut disposer immediatement de plusieurs informations et ceci n'est possible qu'avec des communications.

– Les réponses dynamiques des actionneurs longitudinaux (locomotion et freinage) peuvent réduire considérablement la performance d'un contrôle longitudinal. Une réponse rapide est nécessaire pour donner une bonne précision de la distance entre les véhicules et une stabilité asymptotique, surtout si le train veut garder une distance constante.

– Le choix de l'inter-distance entre les véhicules reste très discutable.

– Le contrôle longitudinal doit être robuste vis à vis des variations des paramètres dans les conditions réelles telles que : les forces externes (pentes, vent, ...), les anomalies des capteurs. Ceci demande un énorme travail expérimental.

– La performance du contrôle longitudinal dépendra de la qualité du capteur de distance. Ce dernier doit être capable de trouver avec certitude le bon véhicule mais aussi détecté les obstacles.

1.3.5 Etat de la recherche sur le contrôle latéral/longitudinal

Bien que le volume de recherche dans le domaine de l'AVCS est concentré soit sur le contrôle longitudinal ou le contrôle latéral, il existe quelques projets qui ont tenté d'intégrer les deux dans un même système de contrôle. Cela va des systèmes d'avertissement

avec un contrôle partiel jusqu'au système entièrement contrôlé, en passant par le contrôle intermittent.

Les systèmes d'avertissement et d'assistance ont été largement étudiés dans le passé par les constructeurs d'automobiles Japonais et Allemand. *Sekine et al.* [STAS94] ont écrit un rapport sur les développements et les tests réalisés par Honda sur un système de conduite qui avertit le conducteur si l'accélération latérale dépasse $0.2\ g$ et si la décélération dépasse $0.4\ g$. *Butsuen et al.* [BDS94] décrivent le système d'évitement d'obstacles de Mazda, qui agit automatiquement sur le système de freinage grâce à un ensemble de donner capteurs composés d'un radar laser pour détecter les obstacles, d'un système de traitement d'image pour reconnaître les lignes blanches et d'autres capteurs pour la reconnaissance de l'état de surface de la route. Bien que le contrôle ne soit que longitudinal (freinage), la fonction de traitement des capteurs permet de donner la position du véhicule par rapport aux lignes (pour le contrôle latèral) et la distance par rapport aux obstacles (pour le contrôle longitudinale).

Dans le cadre du programme PROMETHEUS, les deux sociétés Allemandes Porsche et BMW ont choisi d'assister le conducteur plutôt que de réaliser un contrôle entièrement automatique. *Reichart* a décrit dans [RN94] les travaux de BMW dans ce domaine. L'implémentation consiste à donner un couple et un retour en force au conducteur par l'intermédiaire du volant et de la pédale d'accélération [NR94, RN94]. Cette approche nécessite une modèlisation des réponses du conducteur à ces stimulus, en introduisant des facteurs humains compliqués. Les résultats des tests du système dans un trafic réel ont été très encourageants. *Andrews et al.* [AJRS94] ont décrit les travaux de Porsche pour l'optimisation de l'utilisation des pneus dans la conduite et dans des conditions difficiles. Une méthode acoustique innovatrice pour estimer le contact pneu/sol a été introduite. Cette information peut être utilisée soit pour avertir le conducteur lorsque les pneus atteignent leur limite de frottement, soit donner un retour en force à travers la pédale d'accélération, soit réaliser directement un contrôle de vitesse du véhicule.

Les recherches pour l'intégration des contrôles longitudinal et latéral pour une automatisation complète des véhicules ont été conduit au même temps par les industriels et les universitaires. Dans le milieu universitaire, *Kehtarnavaz* [KLG90] décrit l'utilisation d'un système de traitement d'image pour permettre au véhicule de suivre un autre véhicule équipé avec des cibles lumineuses. *Sheikholeslam* et *Desoer* [SD92a] décrivent une modélisation et une simulation pour combiner un contrôle latéral et longitudinal utilisant des modèles relativement simple sans couplage des mouvements latéraux et longitudinaux. *Pham* [PHT94] étend cette approche sur des modèles et des lois de commandes plus compliquées. *Pomerleau et al.* [PTLR94] ont fait un état de l'art des recherches sur ce contrôle dans l'institut de robotique du CMU (**C**arnegie **M**ellon **U**niversity). Des véhicules terrestres autonomes peuvent faire le suivi de lignes, suivi de véhicule et parking, en utilisant la vision comme capteur exteroceptif. Ces recherches sont orientées plus vers les problèmes d'intelligence artificielle et le traitement d'image que sur les problèmes des dynamiques des véhicules et de contrôle.

Pour le milieu industriel, l'intégration du contrôle longitudinal/latéral pour une automatisation complète des véhicules apparaît avec les nouvelles technologies. Quelques implémentations expérimentales significatives ont été réalisées. Mazda a développé un prototype entièrement automatique utilisant un système de traitement d'image pour le suivi de ligne et la détection d'obstacle. Le système est décrit dans un premier article en 1990 [OKF90], un deuxième plus complet avec les résultats de tests est donné dans [THJ+93]. Des travaux similaires réalisés par Mitsubishi ont aussi été reportés à la fin des années 90 [HY90], en insistant sur le confort des passagers dans le véhicule lors des manœuvres de changement de voie automatique. De son coté, Toyota s'intéresse à la réalisation de la conduite entièrement automatique dans des routes publiques [OKI92, SHW92, OTO92]. Vers 1994, ils ont étendu leurs travaux pour développer un système qui pourrait être appliqué à l'automatisation de la conduite de véhicules dans des routes spéciales (utilisant un câble magnetique, un système de stéréo-vision pour la détection des lignes et un radar laser pour la détection des véhicules) [ST94b]. Probablement, le développement le plus sophistiqué à cette date a été réalisé par Daimler Benz avec le prototype VITA II, qui est une voiture type mercedes équipée avec 12 caméras et des processeurs spécialisés en traitement d'images [Ulm94]. Ce véhicule peut faire le suivi de ligne et le suivi de véhicule, la détection et l'évitement des obstacles. Il a été testé en mode automatique sur l'autoroute en réalisant un parcourt aller-retour Paris-Lille à l'occasion de la présentation des démonstrateurs de PROMETHEUS.

1.3.6 Etat de la recherche sur les problèmes du contrôle haut niveau

Les recherches se sont focalisées sur le contrôle latéral et longitudinal. Cependant pour une automatisation complète, il est nécessaire de traiter les fonctions de haut niveau comme, le calcul des trajectoires à suivre pour optimiser le flot des véhicules automatiques dans le trafic.

Les problèmes de la couche de coordination ont été premièrement traités par *Hsu et al.* [HSEV91]. Ces chercheurs ont divisé en un ensemble de protocole les manœuvres fondamentales que le véhicule peut faire en coopération avec ces voisins (changement de ligne, rejoindre le train, se séparer du train,...). La logique de ces manœuvres (avec la communication inter-véhicules) a été définie sous la forme d'un automate d'état fini pour pouvoir vérifier la correction des manœuvres [P.93]. Les objectifs de l'intégration de la représentation d'événements discrets avec le niveau continu de régulation sont définis par *Lygeros* et *Godbole* [LG94]. *Rao* et *Varaiya* utilisent une simulation pour évaluer les implications des manœuvres de coordinations sur la capacité des lignes dans un système automatisé avec et sans communication entre les véhicules [RVE93b, RVE93a]. *Niehaus* et *Stengel* adoptent une approche s'appuyant sur un système expert pour définir les manœuvres qu'un véhicule autonome peut faire, en se basant sur la perception avec une machine de vision pour choisir les actions adoptées par les véhicules [NS90a, NS90b, NS91]. Contrairement aux papiers précédents, cette étude suppose que la communication n'est pas indispensable. Une approche analogue d'implémentation de règles de code de la route est proposé par *Pilutti*

et al. [PRK90] pour avertir les conducteurs en cas de violation de ces régles.

Au dessus du niveau de coordination on peut trouver le niveau de lien, où des variables sont associées avec le flot de véhicules au lieu de l'être sur les mouvements individuelles de chaque véhicule. Le premier traitement de ce problème est donné par *Karaaslan et al.* dans [KVW90]. Plusieurs lignes, fixées en fonctions des vitesses, sont utilisées afin d'atténuer les perturbations du flot. Ces solutions ont été utilisées par *Rao* et *Varaiya* en ajoutant des heuristiques afin de faciliter l'accès des véhicules dans l'autoroute automatique [RV94a, RV94b]. Ces papiers montrent de très bonnes performances dans des conditions normales et de cas d'accidents. *Ramaswamy et al.* [RMPB94] ont adopté une approche différente pour assigner automatiquement les lignes aux véhicules, en se basant sur une optimisation centralise des trajectoires. Cette approche fait apparaître une communication intense. *Peng* a adopté une approche inspirée de la théorie du contrôle optimal pour traiter le probleme de l'administration du trafic [Pen93].

Le plus haut niveau d'une automatisation complète d'une autoroute, est le niveau réseau. Les études des solutions à ce niveau ont commencé avec le système AGT (Automated Guideway Transit) en 1970. Ce travail a été repris par le groupe OSU [FM91]. Ces travaux sont maintenant considérés comme dépassés parce que d'énormes progrès ont été réalisés dans les technologies des micro-processeurs, les langages de programmation et les techniques de communications. La première tentative d'utilisation d'une approche moderne à ce niveau a été entamée par *Tsao* [Tsa94].

Chapitre 2

Fonctionnalités de Base pour le Contrôle-Commande de Véhicules

COMME on l'a déjà évoqué, la programmation d'une mission nécessite l'utilisation d'un ensemble de lois de commandes. Et l'intégration de ces dernières demande le développement de plusieurs fonctionnalités spécifiques au système robotique et à la mission. Ces fonctionnalités incluent d'une part, le développement des capteurs et d'autre part des algorithmes de traitement de leurs données. Afin de faciliter le travail de l'automaticien, il faut que l'environnement de programmation soit capable de lui fournir toutes les fonctionnalités utiles pour l'intégration de la commande. Par exemple, les mesures des capteurs sont souvent bruitées et il faut pouvoir procéder à un filtrage. D'autre part, la conception de la commande n'est possible que si toutes les variables d'état du système (position, vitesse, accélération, ..) sont physiquement accessibles. Dans la réalité, ce n'est pas toujours vrai. Donc, il faut procéder à une reconstruction de l'état à partir des seules informations données par les capteurs, à savoir les mesures. Un bon environnement doit fournir le maximum de ces fonctions.

Dans ORCCAD ces fonctions sont réalisées dans le code des TMs de la partie réflexe de la TR. L'utilisateur dispose d'une librairie de TMs spécifique à l'architecture de son système robotique, et il n'a que la charge de faire le lien entre les TMs pour construire les actions robotiques nécessaires pour programmer la mission. Dans ce chapitre, on s'intéresse aux algorithmes qui réalisent ces fonctions pour les robots mobiles à roues en générale, et pour le véhicule de type voiture en particulier. Nous avons adopté une approche générique basée sur une modélisation cinématique. Le plan de ce chapitre est les suivant :

- ☞ dans la section 2.1, nous commencerons par présenter une approche de modélisation des robots mobiles à roues évoluant sur une surface plane. La modélisation, inspirée des travaux de Campion et al. [CBAN96], est valable lorsque les hypothèses de roulement sans glissement sont satisfaites. Comme nous allons le détailler, ces hypothèses faites sur chacune des roues rangent les robots mobiles parmi la classe des systèmes non holonomes ;

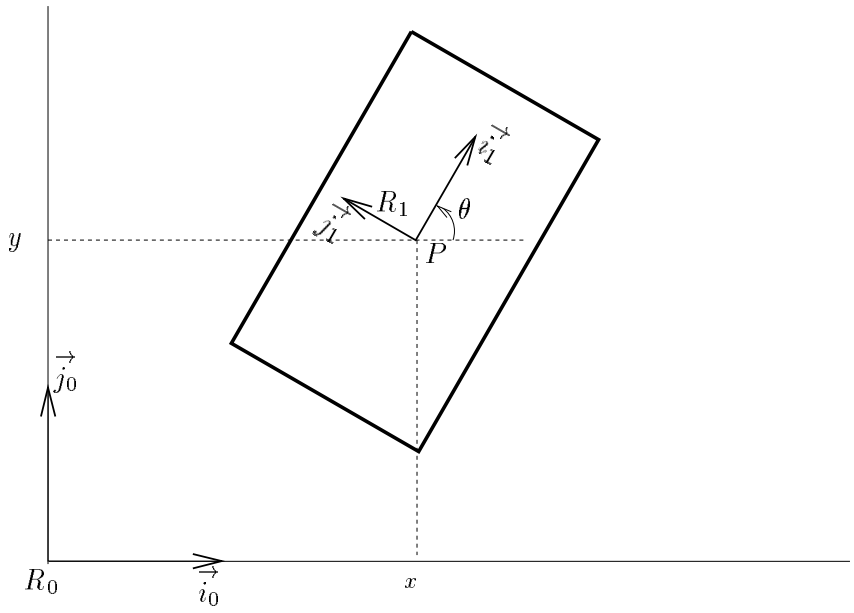


FIG. 2.1-1 – *Position d'une plate-forme dans un référentiel.*

- ☞ dans la section 2.2, l'approche sera appliquée sur le prototype ATALANTE. Dans la même section, et d'après les équations du modèle obtenu, les algorithmes d'estimation de l'état du véhicule à partir des mesures seront détaillés. Les algorithmes obtenus seront intégrés dans la librairie classe AQ-PR des TMs ;

- ☞ nous présenterons, dans la section 2.3, les deux capteurs extéroceptifs développés dans le projet PRAXITÈLE pour satisfaire les objectifs de la mission. Le premier permet de déterminer la position relative d'un véhicule par rapport à un autre : il sera utilisé dans le suivi de véhicule. Le deuxième permet de faire une reconstruction de l'environnement : il sera utilisé dans la manœuvre de parking parallèle. Les données de ces deux capteurs seront accéssibles directement par l'utilisateur d'ORCCAD grâce aux TMs de la classe AQ-EX.

2.1 Modélisation cinématique générique des RMs à Roues

2.1.1 Les robots mobiles considérés

Les robots mobiles considérés sont constitués d'une plate-forme et de plusieurs roues liées mécaniquement à cette dernière. La position du robot dans un plan horizontal est montrée dans la figure 2.1-1. Soit le repère inertiel $R_0(O, \vec{i}_0, \vec{j}_0)$ et soit un point de référence P situé sur la plate-forme que l'on prend comme centre du repère mobile $R_1(P, \vec{i}_1, \vec{j}_1)$. La position du robot est complètement défini par l'utilisation des trois variables x, y, θ où :

- x, y sont les coordonnées d'un point de la plate-forme P, exprimés dans le repère fixe R_0 : $\overrightarrow{OP} = x\vec{i}_0 + y\vec{j}_0$

– θ est l'orientation du repère R_1 par rapport au repère fixe R_0.

A l'aide de ces variables, on définit le vecteur de posture, ξ,de la base mobile et la matrice de rotation $R(\theta)$:

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{2.1-1}$$

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1-2}$$

Le mouvement du robot mobile sur le plan horizontal est complètement définie par les variables donnant la position et l'orientation de la plate forme (c'est à dire la posture) et les variables de rotations des différentes roues et de leurs vitesses dans le temps. L'hypothèse de roulement sans glissement, posée sur les différentes roues, impose sur le système des contraintes. C'est la nature de ces contraintes que nous allons détailler dans le prochain paragraphe.

2.1.2 Contraintes non holonomes

Une liaison résulte d'un contact entre deux solides, et est donc caractérisée par la nature géométrique et la nature physique du contact. Mathématiquement, elle se traduit par des égalités (ou des inégalités) entre les paramètres du système, les vitesses généralisées et le temps.

Lorsque les solides ne sont astreints qu'à des conditions géométriques de contact entre eux, les relations de liaison ne font pas intervenir les vitesses. Ces liaisons sont dites *holonomes* ou *géométriques*, et sont alors de la forme : $f(q,t) = 0$.

Lorsqu'en plus des conditions géométriques de contact, les solides d'un système sont astreints à certaines conditions cinématiques de contact, il en résulte des relations de liaison qui font intervenir les vitesses généralisées. Les liaisons correspondantes sont dites *non holonomes* ou *cinématiques* lorsque son expression : $f(q,\dot{q},t) = 0$ est non intégrable.

Notons qu'une liaison non holonome nécessite au préalable une liaison holonome traduisant le contact purement géométrique.

2.1.3 Contraintes sur la vitesse d'une roue

Considérons la liaison d'une roue sur une plate-forme ayant les caractéristiques géométriques indiquées sur la figure 2.1-2. La roue est orientable et tourne autour du point A. La vitesse au point de contact C de la roue avec le sol exprimée dans le repère R_{O_r} est donnée par (voir démonstration en Annexe A) :

$$V^r_{C/O} = \begin{bmatrix} -\sin(\alpha+\beta-\delta) & \cos(\alpha+\beta-\delta) & l\cos(\beta-\delta)+d\cos\delta \\ -\cos(\alpha+\beta-\delta) & -\sin(\alpha+\beta-\delta) & l\sin(\delta-\beta)+d\sin\delta \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} d\dot{\beta}\cos\delta - r\dot{\varphi} \\ d\dot{\beta}\sin\delta \end{bmatrix} \tag{2.1-3}$$

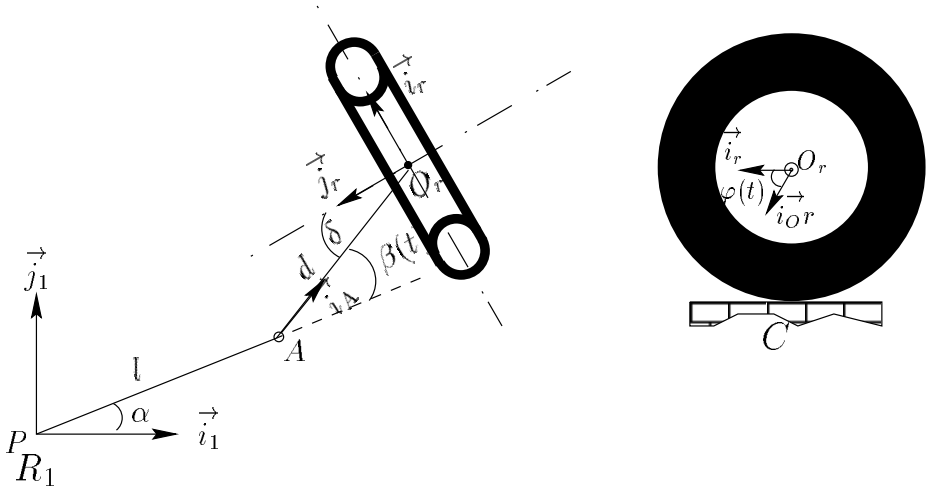


FIG. 2.1-2 – *Description d'une roue.*

La contrainte de roulement sans glissement est traduite par la condition $V^r_{C/O} = 0$. Cette expression exprime deux contraintes suivant les deux composantes de cette vitesse :

- suivant le plan de la roue $\left(V^r_{C/O}\right)_{\vec{i}_r} = 0$. La première équation de (2.1-3) exprime la condition de rotation pure, ou de non glissement longitudinal,
- suivant le plan orthogonal à celui formé par la roue $\left(V^r_{C/O}\right)_{\vec{j}_r} = 0$. La deuxième ligne de l'équation (2.1-3) traduit la condition de non glissement latéral.

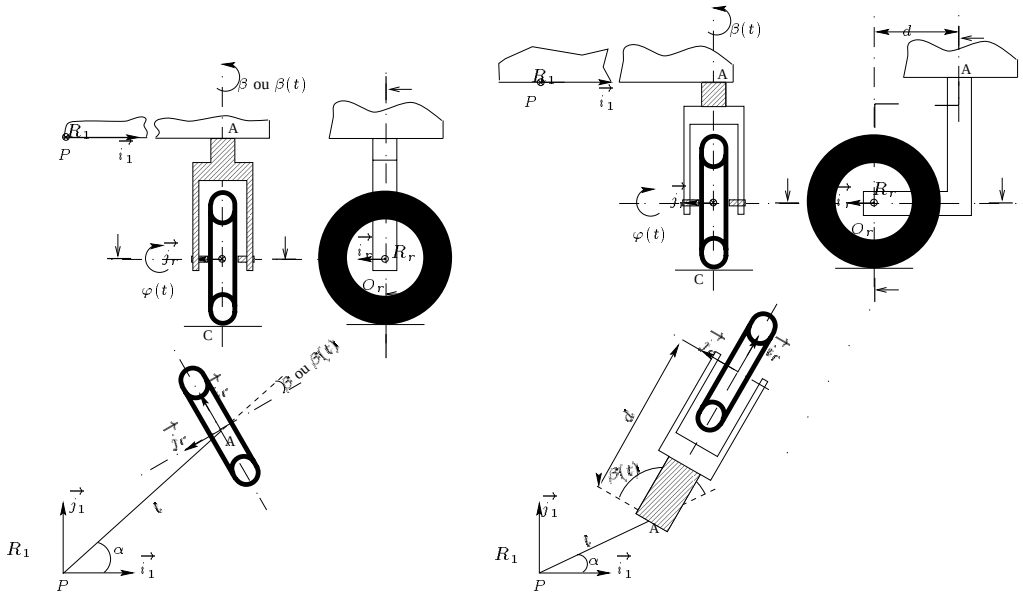
Suivant les paramètres δ, d et $\dot{\beta}$, nous pouvons distinguer cinq types de roues :

La roue fixe La roue est dite "fixe" lorsque son orientation reste constante : $\dot{\beta} = 0$, $\delta = -\pi/2$, $d = 0$ (voir figure 2.1-3(a)). Les deux contraintes de non glissement s'expriment par :

$$\begin{bmatrix} \left(V^r_{C/O}\right)_{\vec{i}_r} \\ \left(V^r_{C/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha+\beta) & -\sin(\alpha+\beta) & -l\sin(\beta) \\ \sin(\alpha+\beta) & -\cos(\alpha+\beta) & -l\sin(\beta) \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ 0 \end{bmatrix} = 0 \tag{2.1-4}$$

La roue centrée orientable La roue est orientable donc $\dot{\beta} \neq 0$ (voir figure 2.1-3(a)). On remplaçant $\delta = -\pi/2$, $d = 0$, et $\dot{\beta} \neq 0$ dans l'équation (2.1-3). Les deux contraintes de non glissement deviennent :

$$\begin{bmatrix} \left(V^r_{C/O}\right)_{\vec{i}_r} \\ \left(V^r_{C/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha+\beta) & -\sin(\alpha+\beta) & -l\sin(\beta) \\ \sin(\alpha+\beta) & -\cos(\alpha+\beta) & -l\sin(\beta) \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ 0 \end{bmatrix} = 0 \tag{2.1-5}$$



(a) Roue fixe ou centrée orientable.

(b) Roue décentrée orientable 1.

(c) Roue décentrée orientable 2.

FIG. 2.1-3 – *Classification des roues.*

La roue décentré orientable 1 Cette roue, appelée aussi roue folle diffère de la précédente par la présence du corps de longueur d (voir figure 2.1-3(b)). Dans ce cas, $\delta = -\pi/2$, $d \neq 0$, $\beta \neq 0$ et les contraintes de non glissement s'écrivent :

$$\begin{bmatrix} \left(V^r_{C/O}\right)_{\vec{i}_r} \\ \left(V^r_{C/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -\cos(\alpha+\beta) & -\sin(\alpha+\beta) & -l\sin(\beta) \\ \sin(\alpha+\beta) & -\cos(\alpha+\beta) & -l\sin(\beta)-d \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ -d\dot{\beta} \end{bmatrix} = 0 \tag{2.1-6}$$

La roue décentré orientable 2 Contrairement à la roue précédente, le mouvement du roulement de cette roue est effectué perpendiculairement au corps d, donc $\delta = 0$ (voir figure 2.1-3(d)). Les contraintes de non glissement sont :

$$\begin{bmatrix} \left(V^r_{C/O}\right)_{\vec{i}_r} \\ \left(V^r_{C/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -\sin(\alpha+\beta) & \cos(\alpha+\beta) & l\cos(\beta)+d \\ -\cos(\alpha+\beta) & -\sin(\alpha+\beta) & l\cos(\beta) \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} d\dot{\beta} - r\dot{\varphi} \\ 0 \end{bmatrix} = 0 \tag{2.1-7}$$

La roue suédoise La position de cette roue est décrite de la même façon que la roue fixe, c'est à dire grâce aux trois constantes : α, β et l, plus un nouveau paramètre, l'angle γ, qui caractérise la direction où la composante de la vitesse latérale est nulle. Ce qui réduit les contraintes de glissement à une seule équation :

$$\left[\left(V^r_{C/O}\right)_{\vec{i}_r} \right] = \left[-\sin(\alpha+\beta+\gamma) \quad \cos(\alpha+\beta+\gamma) \quad l\cos(\beta+\gamma) \right] R(\theta)\ \dot{\xi} + r\cos\gamma\ \dot{\varphi} = 0 \tag{2.1-8}$$

2.1.4 Modélisation générique proposée par Campion et al

Pour déterminer le modèle cinématique d'un robot mobile à roue, Campion et al. [CBAN96], considèrent un robot mobile doté de N roues de types fixes, centrées orientables, décentrées orientables 1 et suédoises ($N = N_f + N_c + N_{oc} + N_{sw}$). La configuration du robot est entièrement décrite par :

- la posture ξ,
- les coordonnées angulaires $\beta_c(t)$ et $\beta_{oc}(t)$,
- le vecteur de rotation des roues autour de leur axe de rotation :
  $\varphi(t) = \left[\varphi_f(t) \quad \varphi_c(t) \quad \varphi_{oc}(t) \quad \varphi_{sw}(t) \right]^T$

Le nombre total des coordonnées de configuration est : $N = N_f + 2N_c + 2N_{oc} + N_{sw} + 3$. Les contraintes cinématiques de non glissement peuvent être écrites sous la forme matricielle suivante :

$$\begin{array}{c} J_1(\beta_c, \beta_{oc}) R(\theta)\ \dot{\xi} + J_2\ \dot{\varphi} = 0 \quad \text{rotation pure} \\ C_1(\beta_c, \beta_{oc}) R(\theta)\ \dot{\xi} + C_2\ \dot{\beta}_{oc} = 0 \quad \text{non glissement lateral} \end{array} \tag{2.1-9}$$

avec : $J_1(\beta_c, \beta_{oc}) = \left[J_{1f} \quad J_{1c}(\beta_c) \quad J_{1oc}(\beta_{oc}) \quad J_{1sw} \right]^T$, $J_2 = diag(r, r\cos\gamma)$ où J_{1f}, J_{1c},J_{oc} et J_{sw} sont des matrices de dimension respective $(N_f \times 3), (N_c \times 3), (N_{oc} \times 3)$ et $(N_{sw} \times 3)$ issues des équations de rotation pure (2.1-4), (2.1-5), (2.1-6) et (2.1-8). J_2 est une matrice diagonale de dimension $(N \times N)$ où les termes diagonaux sont les rayons des roues exceptés pour les roues suédoises où le rayon est multiplié par $\cos\gamma$.

$C_1(\beta_c, \beta_{oc}) = \begin{bmatrix} C_{1f} & C_{1c}(\beta_c) & C_{1oc}(\beta_{oc}) \end{bmatrix}^T, C_2 = \begin{bmatrix} 0 & 0 & C_{2oc} \end{bmatrix}^T$ où C_{1f}, C_{1c} et C_{oc} sont des matrices de dimension $(N_f \times 3), (N_c \times 3)$ et $(N_{oc} \times 3)$ issues des équations de rotation pure (2.1-4), (2.1-5) et (2.1-6). C_{2oc} est une matrice diagonale constante où les termes diagonaux représentent les distances d des roues décentrées orientables 1.

2.1.5 Classification des robots mobiles suivant la mobilité

Si on considère les $(N_f + N_c)$ premières contraintes de l'équation (2.1-9), c'est à dire les contraintes de non glissement pour les roues fixes et les roues centrées, on obtient alors :

$$\begin{array}{c} C_{1f} R(\theta)\, \dot{\xi} = 0 \\ C_{1c}(\beta_c)\, R(\theta)\, \dot{\xi} = 0 \end{array} \tag{2.1-10}$$

Si on pose : $C_1^*(\beta_c) = \begin{bmatrix} C_{1f} & C_{1c}(\beta_c) \end{bmatrix}^T$
L'équation (2.1-10) devient :

$$C_1^*(\beta_c)\, R(\theta)\, \dot{\xi} = 0 \tag{2.1-11}$$

Cette dernière équation représente les contraintes non holonomes imposées par les roues fixes et les roues centrées du robot. Elles traduisent le faite que le robot peut évoluer dans un espace, de dimension 3, associée à la posture ξ avec des restrictions sur l'évolution de la posture dans le temps.

$$C_1^*(\beta_c)\, R(\theta)\, \dot{\xi} = 0 \Rightarrow R(\theta)\, \dot{\xi} \in Ker[C_1^*(\beta_c)]$$

Deux grandeurs sont définies par Campion, Bastin et d'Andréa-Novel pour faire la classification :

$$\begin{array}{c} Degré\ de\ mobilité\ : \delta_m = \dim\{Ker\,[C_1^*(\beta_c)]\} = 3 - rang\,[C_1^*(\beta_c)] \\ Nombre\ de\ roues\ indépendament\ orientables\ : \delta_s \end{array}$$

Il en résulte cinq types de robots mobiles selon le couple d'indice (δ_m, δ_s).

- 1 robot mobile omnidirectionnel : $(3, 0)$;
- 4 robots mobiles à mobilité restreinte : $(2, 0)$, $(1, 1)$, $(2, 1)$, $(1, 2)$.

Remarque 2.1.1 *Si le* $rang\,[C_1^*(\beta_c)] = 3$ *alors* $R(\theta)\, \dot{\xi} = 0$*, c'est à dire que la plate-forme de la voiture ne peut faire aucun mouvement dans le plan. Donc pour que le robot puisse bouger, il faut que :* $rang\,[C_1^*(\beta_c)] \leq 2$.

2.1.6 Le modèle cinématique de posture

Reprenons l'équation (2.1-11) donnant les contraintes sur les vitesses latérales nulles. Le vecteur $R(\theta)\,\dot{\xi}$ appartient au noyau engendré par $C_1^*(\beta_c)$ (ou bien $R(\theta)\,\dot{\xi} \in Ker[C_1^*(\beta_c)]$). Par conséquent :

quelque soit t, il existe un vecteur $\eta(t)$ tel que :

$$\dot{\xi} = R(\theta)^T \sum(\beta_c)\eta \tag{2.1-12}$$

où les colonnes de $\sum(\beta_c)$ forment une base de $Ker[C_1^*(\beta_c)]$, c'est à dire : $C_1^*(\beta_c)\sum(\beta_c) = 0$.

Si le robot n'a pas de roue centrée orientable $(N_c = 0)$, la matrice $\sum$ est constante. Dans le cas contraire $(N_c \geq 1)$, la matrice $\sum$ dépend des angles β_c. On complète l'expression (2.1-12) comme suit :

$$\begin{array}{c} \dot{\xi} = R(\theta)^T \sum(\beta_c)\eta \\ \dot{\beta}_c = \zeta \end{array} \tag{2.1-13}$$

Ces équations sont les représentations d'état du modèle cinématique de posture où les coordonnées de la posture ξ et les coordonnées angulaires β_c (qui existe pour $N_c \geq 1$) sont les variables d'états. Et où η et ζ peuvent être interprétées comme étant les variables de commande (homogènes à des vitesses). Ce modèle décrit parfaitement l'évolution de la plate-forme. Mais il est incomplet car l'évolution des vitesses de rotations $\dot{\varphi}$ et, si elle existe, la vitesse angulaire $\dot{\beta}_{oc}$n'interviennent pas. En effet, nous avons tenu compte que des équations décrivant les contraintes de non glissement latérale des roues fixes et des roues centrées orientables 1. La prochaine section va compléter ce modèle en prenant en compte l'équation de la contrainte de glissement longitudinal.

2.1.7 Le modèle cinématique de configuration

D'une part, la première équation (2.1-9), donnant les contraintes sur les vitesses longitudinales nulles aux points de contact des différentes roues, permet de donner l'évolution de $\dot{\varphi}$:
$\dot{\varphi} = -J_2^{-1} J_1(\beta_c, \beta_{oc}) R(\theta)\,\dot{\xi}$, et d'autre part, l'équation (2.1-9), donnant les contraintes sur les vitesses latérales nulles aux points de contact des différentes roues, permet de donner l'évolution de $\dot{\beta}_{oc}$: $\dot{\beta}_{oc} = -C_{2oc}^{-1} C_{1oc}(\beta_{oc}) R(\theta)\,\dot{\xi}$

Or l'équation du modèle cinématique de posture (2.1-13), nous donne : $R(\theta)\,\dot{\xi} = \sum(\beta_c)\eta$

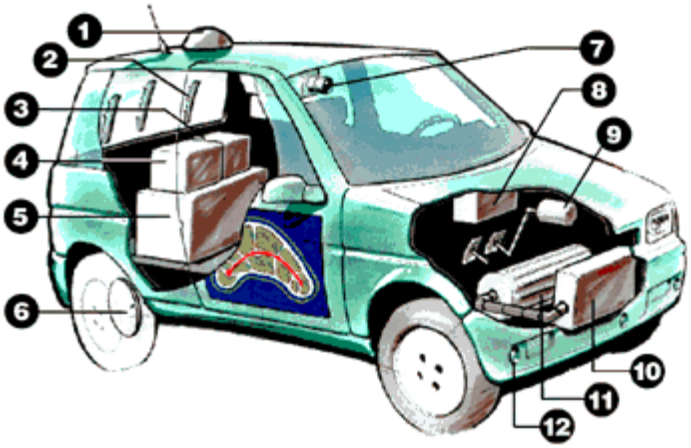
Donc les équation d'états décrivant $\dot{\beta}_{oc}$ et $\dot{\varphi}$ deviennent :

$$\begin{array}{l} \dot{\beta}_{oc} = -C_{2oc}^{-1} C_{1oc}(\beta_{oc}) \sum(\beta_c)\eta \\ \dot{\varphi} = -J_2^{-1} J_1(\beta_c, \beta_{oc}) \sum(\beta_c)\eta \end{array}$$

Soit q le vecteur de configuration du robot : $q = \begin{pmatrix} \xi & \beta_c & \beta_{oc} & \varphi \end{pmatrix}^T$, le modèle cinématique complet appelé *modèle cinématique de configuration* s'écrira : $\dot{q} = S(q)u$

$$\text{où : } S(q) = \begin{pmatrix} R(\theta)^T \sum(\beta_c) & 0 \\ 0 & I \\ -C_{2oc}^{-1} C_{1oc}(\beta_{oc}) \sum(\beta_c) & 0 \\ -J_2^{-1} J_1(\beta_c, \beta_{oc}) \sum(\beta_c) & 0 \end{pmatrix} \text{ et } u = \begin{pmatrix} \eta \\ \zeta \end{pmatrix}$$

2.2 Modélisation du prototype ATALANTE



1- Antenne GPS.
2- Balises infrarouges.
3- Ordinateur de vision (PC + 2 cartes de traitement d'image).
4- Ordinateur de localisation (PC).
5- Ordinateur principal (VME + 4 cartes filles).
6- Codeur incrémental.
7- Caméra linéaire.
8- Direction électrique avec un codeur.
incrémental et un potentiomètre.
9- Servo-frein électrique avec un capteur de pression.
10- Commande électronique de puissance.
11- Moteur de puissance avec un codeur incrémental.
12- Capteurs ultrasonores.

Nous allons à présent appliqué l'approche présentée sur le prototype ATALANTE. Son schéma 2D est représenté à la figure 2.2-4. C'est une voiture constituée :

- d'une plate-forme ;
- de 2 roues fixes (roue 3 et roue 4) ;
- de 2 roues centrées orientables 2 (roue 1 et roue 2).

L'évolution de la voiture sur une surface plane est complètement décrite par les variables suivantes :

- les coordonnées (x, y) exprimées dans le repère inertiel R_0, d'un point P de la plate-forme, choisie au milieu de l'essieu arrière ;
- l'orientation θ de la plate-forme, à laquelle on associe le repère R_1 ;
- respectivement les angles d'orientations β_1, β_2 des roues 1, 2 ;
- les angles de rotation $\varphi_i, i = 1, ..., 4$ des 4 roues.
- r, l_1 et l_2 étant respectivement le rayon d'une roue, la largeur des essieux et la distance entre les 2 essieux.

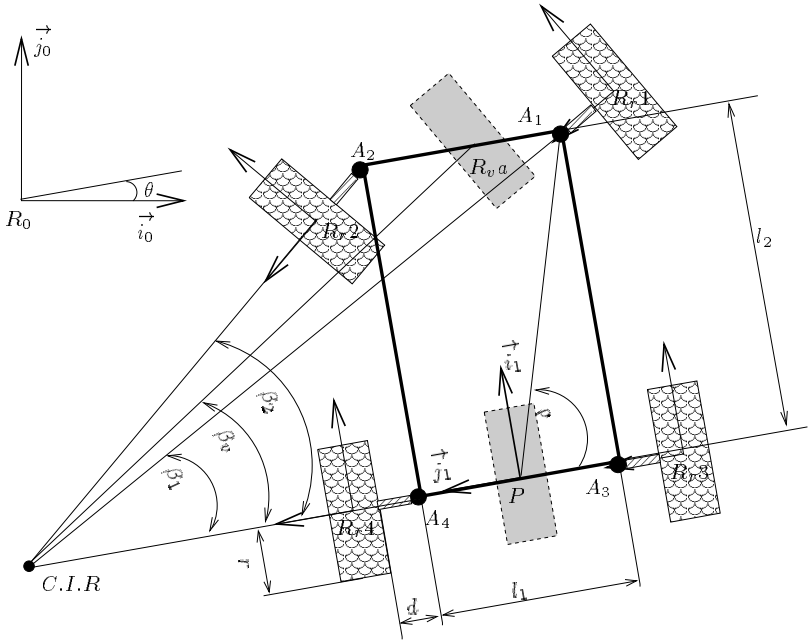


FIG. 2.2-4 – *Schéma 2D du prototype.*

Nous ajoutons deux autres roues virtuelles qui, comme on va le montrer par la suite, permettrons une écriture simplifiée et concise de notre modèle. La première fixe est située au milieu de l'essieu arrière (caractérisée par son angle de rotation φ), la deuxième est centrée orientable au milieu de l'essieu avant (caractérisée par φ_m et β_v).

Repères

- R_0 repère inertiel ;
- R_1 repère associé à la plate-forme ayant comme origine le point P (centre de l'essieu arrière) ;
- $R_{ri}, i = 1, ..., 4,$ les repères attachés aux structures des 4 roues et ayant comme origine le centre de rotation ;
- $R_{Ai}, i = 1, ..., 4,$ les repères attachés aux leviers des 4 roues et ayant comme origine le centre d'orientation.

2.2.1 Les vitesses aux points de contacts

En appliquant la formule (2.1-3) aux quatre roues, on obtient :

Roue 1 $\delta = 0 \,;\, d = d \,;\, \alpha + \beta = \beta_1 - \pi/2 \,;\, \beta = \beta_1 - \rho \,;\, \varphi = \varphi_1.$

$$\begin{bmatrix} \left(V^{r_1}_{C_1/O}\right)_{\vec{i_r}} \\ \left(V^{r_1}_{C_1/O}\right)_{\vec{j_r}} \end{bmatrix} = \begin{bmatrix} \cos\beta_1 & \sin\beta_1 & l_1\cos\beta_1 + l_2\sin\beta_1 + d \\ -\sin\beta_1 & \cos\beta_1 & -l_1\sin\beta_1 + l_2\cos\beta_1 \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} d\dot{\beta}_1 - r\dot{\varphi}_1 \\ 0 \end{bmatrix}$$

Roue 2 $\delta = \pi\,;d = d\,;\alpha+\beta = \beta_2+\pi/2\,;\beta = \beta_2-\rho\,;\varphi = \varphi_2$.

$$\begin{bmatrix} \left(V^{r_2}_{C_2/O}\right)_{\vec{i}_r} \\ \left(V^{r_2}_{C_2/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} \cos\beta_2 & \sin\beta_2 & -l_1\cos\beta_2 - l_2\sin\beta_2 - d \\ -\sin\beta_2 & \cos\beta_2 & l_1\sin\beta_2 - l_2\cos\beta_2 \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -d\dot{\beta}_2 - r\dot{\varphi}_2 \\ 0 \end{bmatrix}$$

Roue 3 $\delta = 0\,;d = d\,;\alpha = 3\pi/2;\beta = 0\,;\varphi = \varphi_3$.

$$\begin{bmatrix} \left(V^{r_3}_{C_3/O}\right)_{\vec{i}_r} \\ \left(V^{r_3}_{C_3/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & l_1 + d \\ 0 & 1 & 0 \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi}_3 \\ 0 \end{bmatrix}$$

Roue 4 $\delta = \pi\,;d = d\,;\alpha = \pi/2\,;\beta = 0\,;\varphi = \varphi_4$.

$$\begin{bmatrix} \left(V^{r_4}_{C_4/O}\right)_{\vec{i}_r} \\ \left(V^{r_4}_{C_4/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -l_1 - d \\ 0 & 1 & 0 \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi}_4 \\ 0 \end{bmatrix}$$

Roue virtuelle avant $\delta = -\pi/2\,;d = 0\,;\alpha = 0\,;\beta = \beta_v\,;\varphi = \varphi_m$.

$$\begin{bmatrix} \left(V^{r_{va}}_{C_{va}/O}\right)_{\vec{i}_r} \\ \left(V^{r_{va}}_{C_{va}/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -\cos\beta_v & -\sin\beta_v & -l_2\sin\beta_v \\ \sin\beta_v & -\cos\beta_v & -l_2\cos\beta_v \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi}_m \\ 0 \end{bmatrix}$$

Roue virtuelle arrière $\delta = -\pi/2\,;d = 0\,;\alpha = 0\,;\beta = 0\,;\varphi = \varphi$.

$$\begin{bmatrix} \left(V^{r_{vr}}_{C_{vr}/O}\right)_{\vec{i}_r} \\ \left(V^{r_{vr}}_{C_{vr}/O}\right)_{\vec{j}_r} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ 0 \end{bmatrix}$$

2.2.2 Contraintes cinématiques

La condition de *non glissement longitudinal* du système est : $\left(V^{r_1}_{C_1/O}\right)_{\vec{i}_r} = \left(V^{r_2}_{C_2/O}\right)_{\vec{i}_r} = \left(V^{r_3}_{C_3/O}\right)_{\vec{i}_r} = \left(V^{r_4}_{C_4/O}\right)_{\vec{i}_r} = 0$. Elle peut être exprimée sous la forme matricielle suivante :

$$\begin{bmatrix} \cos\beta_1 & \sin\beta_1 & l_1\cos\beta_1 + l_2\sin\beta_1 + d \\ \cos\beta_2 & \sin\beta_2 & -l_1\cos\beta_2 - l_2\sin\beta_2 - d \\ 1 & 0 & l_1 + d \\ 1 & 0 & -l_1 - d \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} d\dot{\beta}_1 - r\dot{\varphi}_1 \\ -d\dot{\beta}_2 - r\dot{\varphi}_2 \\ -r\dot{\varphi}_3 \\ -r\dot{\varphi}_4 \end{bmatrix} = 0 \quad \text{(2.2-14)}$$

La condition de *non glissement latéral* du système est : $\left(V^{r_1}_{C_1/O}\right)_{\vec{j}_r} = \left(V^{r_2}_{C_2/O}\right)_{\vec{j}_r} = \left(V^{r_3}_{C_3/O}\right)_{\vec{j}_r} = \left(V^{r_4}_{C_4/O}\right)_{\vec{j}_r} = 0$

$$\begin{bmatrix} -\sin\beta_1 & \cos\beta_1 & -l_1\sin\beta_1 + l_2\cos\beta_1 \\ -\sin\beta_2 & \cos\beta_2 & -l_1\sin\beta_2 - l_2\cos\beta_2 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} R^T(\theta)\dot{\xi} = 0 \quad \text{(2.2-15)}$$

2.2.3 Modèle de posture de la voiture

Les deux contraintes sur les vitesses latérales des roues virtuelles avant et arrière peuvent être exprimées comme suit :

$$C_1^* (\beta_v) R^T (\theta) \dot{\xi} = 0$$

avec : $C_1^* (\beta_v) = \begin{bmatrix} \sin\beta_v & -\cos\beta_v & -l_2\cos\beta_v \\ 0 & 1 & 0 \end{bmatrix}$ et $\sum(\beta_v) = \begin{bmatrix} l_2\cos\beta_v & l_2\cos\beta_v \\ 0 & 0 \\ \sin\beta_v & \sin\beta_v \end{bmatrix}$

Donc *quelque soit t, il existe un vecteur* $\eta(t)$ *tel que :* $\dot{\xi} = R(\theta)\sum(\beta_v)\eta$.

η matérialise un degré de liberté. La matrice $\sum$ dépend de l'angle d'orientation β_v. L'équation précédente peut être augmentée comme suit :

$$\begin{aligned} \dot{\xi} &= R(\theta)\sum(\beta_v)\eta \\ \dot{\beta}_v &= \zeta \end{aligned}$$

η et ζ peuvent être interprétées comme étant les variables de commande (homogène à des vitesses) appliquées à la roue virtuelle avant.

2.2.4 Modèle cinématique de configuration

Le modèle de posture est complété en utilisant les deux autres contraintes (vitesses longitudinales nulles aux points de contact entre les deux roues virtuelles). Ce qui permet de donner l'évolution de $\dot{\varphi}$ et $\dot{\varphi}_m$:

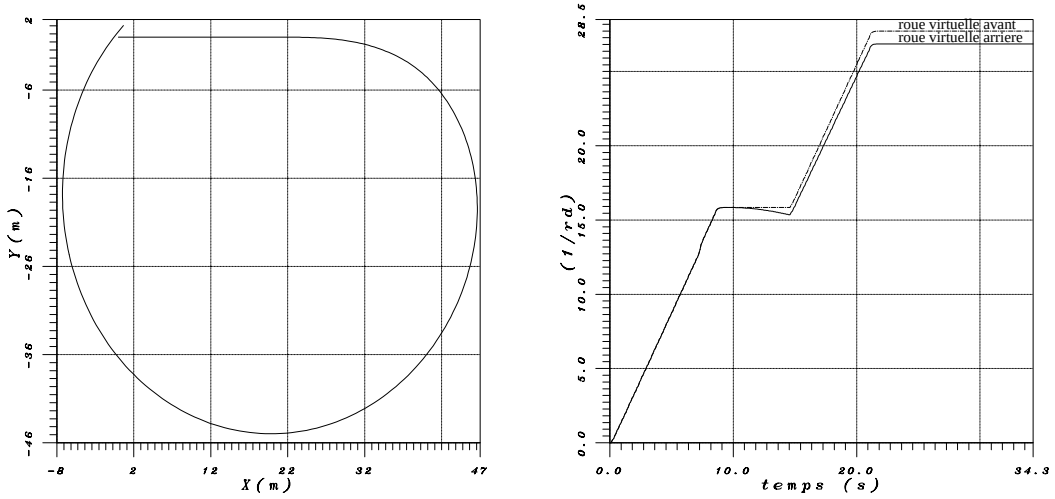
$$r \begin{pmatrix} \dot{\varphi} \\ \dot{\varphi}_m \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ -\cos\beta_v & -\sin\beta_v & -l_2\sin\beta_v \end{bmatrix} R^T(\theta)\dot{\xi}$$

Le modèle cinématique de configuration est :

$$\begin{aligned} \dot{\xi} &= R(\theta)\sum(\beta_v)\eta \\ \dot{\beta}_v &= \zeta \\ \begin{pmatrix} \dot{\varphi} \\ \dot{\varphi}_m \end{pmatrix} &= \frac{l_2}{r} \begin{bmatrix} -\cos\beta_v & -\cos\beta_v \\ -1 & -1 \end{bmatrix} \eta \end{aligned} \qquad \text{(2.2-16)}$$

Ce modèle aurait pu être obtenu de manière quasi-immédiate en utilisant les relations fondamentales de la cinématique. Mais l'approche décrite est systématique et permet d'envisager l'automatisation du processus de modélisation des robots mobiles à roues.

Simulation Nous avons simulé le modèle donné par les équations (2.2-16). Etant donnée une trajectoire (figure 2.2-5(a)), on voit bien que les vitesses des roues virtuelles avant et arrière sont égales sur lorsque la voiture effectue une ligne droite et sont différentes dans les virages (figure 2.2-5(b)).



(a) Trajectoire de la voiture.

(b) Les vitesses des roues virtuelles.

FIG. 2.2-5 – *Simulation de modèle de configuration.*

2.2.5 Modèle de rotation des roues arrières

Les équations de non glissement longitudinale des roues arrières $\left(V^{r_3}_{C_3/O}\right)_{\vec{i}_r} = \left(V^{r_4}_{C_4/O}\right)_{\vec{i}_r} = 0$ expriment la même contrainte donnée par la roue virtuelle arrière $\left(V^{r_{vr}}_{C_{vr}/O}\right)_{\vec{i}_r} = 0$. Par suite :

$$\begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & l_1+d \\ 1 & 0 & -l_1-d \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ -r\dot{\varphi}_3 \\ -r\dot{\varphi}_4 \end{bmatrix} = 0$$

On déduit que :

$$\dot{\varphi} = \frac{\dot{\varphi}_3 + \dot{\varphi}_4}{2} \qquad et \qquad \dot{\theta} = \frac{r(\dot{\varphi}_3 - \dot{\varphi}_4)}{2(l_1+d)} \tag{2.2-17}$$

Si r est le rayon d'une roue. L'équation (2.2-17) devient :

$$V = \frac{r(\dot{\varphi}_3 + \dot{\varphi}_4)}{2} \qquad et \qquad \dot{\theta} = \frac{r(\dot{\varphi}_3 - \dot{\varphi}_4)}{2(l_1+d)} \tag{2.2-18}$$

Remarque 2.2.1 *Ce n'est pas la seule manière d'obtenir les équation (2.2-18), on peut les obtenir on utilisant les équations de non glissement longitudinal de la roue virtuelle arrière* $\left(\left(V^{r_v}_{C_v/O\vec{i}_r}\right)_{\vec{i}_r} = 0\right)$*, l'équation d'une des roues arrières* $\left(\left(V^{r_3}_{C_3/O}\right)_{\vec{i}_r} = 0\right)$ *et l'équation de la roue virtuelle avant* $\left(\left(V^{r_{va}}_{C_{va}/O}\right)_{\vec{i}_r} = 0\right)$

$$\begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & l_1+d \\ -\cos\beta_v & -\sin\beta_v & -l_2\cos\beta_v \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} -r\dot{\varphi} \\ -r\dot{\varphi}_3 \\ -r\dot{\varphi}_m \end{bmatrix} = 0$$

Influence de la différence des rayons En pratique on n'a pas le même rayon $(r_3 \neq r_4)$ car le rayon varie en fonction des déformations pneumatiques. L'équation (2.2-18) devient :

$$V = \frac{r_3\dot{\varphi}_3 + r_4\dot{\varphi}_4}{2} \qquad et \qquad \dot{\theta} = \frac{r_3\dot{\varphi}_3 - r_4\dot{\varphi}_4}{2(l_1+d)} \tag{2.2-19}$$

Soient $r_m = \frac{r_3+r_4}{2}$ le rayon moyen des roues 3 et 4, et $dr = \frac{r_4-r_3}{2}$ la variation de ce rayon. Posons $\dot{\varphi}^+_{34} = \frac{\dot{\varphi}_3+\dot{\varphi}_4}{2} = \dot{\varphi}$ et $\dot{\varphi}^-_{34} = \frac{\dot{\varphi}_4-\dot{\varphi}_3}{2}$. L'équation (2.2-19) s'écrira :

$$V = \dot{\varphi}^+_{34} r_m + \dot{\varphi}^-_{34} dr \qquad et \qquad \dot{\theta} = \frac{\dot{\varphi}^-_{34} r_m}{(l_1+d)} + \frac{\dot{\varphi}^+_{34} dr}{(l_1+d)} \tag{2.2-20}$$

Si on néglige la différence entre le rayon gauche et le rayon droit dr_{34}, on fausse la mesure de $\dot{\theta}$ car le terme $\frac{\dot{\varphi}^+_{34} dr}{(l_1+d)}$ peut être grand. Par contre l'influence de ce terme sur la mesure de V reste faible.

Simulations Dans la première simulation une trajectoire a été donnée à la voiture (figure 2.2-6) en considéré que $r_3 = r_4$. Nous voyons bien qu'il n'y a pas de différence entre la vitesse réelle (figure 2.2-6(b)) et la vitesse obtenue par odométrie (figure 2.2-6(c)). Ainsi pour le cap réel (figure 2.2-6(b)) et le cap obtenu par odométrie (figure 2.2-6(c)).

Une deuxième simulation avec $dr = 1mm$ a été réalisée (figure 2.2-7). Pendant les 30 premières mètres la voiture effectue une ligne droite (figure 2.2-7(a)), le cap de la voiture, donné par l'odométrie commence à dériver par rapport au cap réel (figure 2.2-7(c)). Par contre, la vitesse donnée par l'odométrie reste sensiblement égale à la vitesse réel (figure 2.2-7(b)).

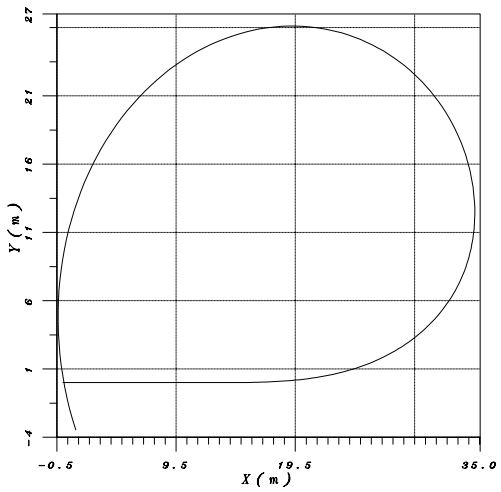
Odométrie

L'odométrie est une méthode de localisation dite *à l'estime* où l'on détermine la position courante (position et cap de la voiture) par intégration des déplacements successifs depuis la position de départ. Les informations sont fournies par deux codeurs incrémentaux placés sur les deux roues arrières.

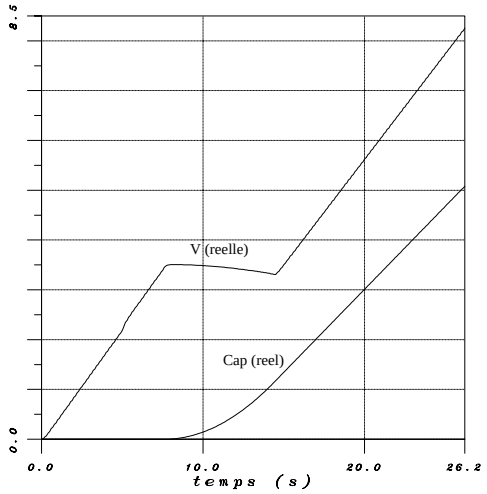
Le nombre de tops, n, donné par un codeur incrémental entre l'instant 0 et l'instant T_e est :

$n =$ Partie entière de $\left(\frac{N\int_0^{T_e}\dot{\varphi}\,dt}{2\pi}\right)$ où $\dot{\varphi}$: représente la vitesse de rotation et N est une caractéristique du codeur (nombre de tops codeur par tour).

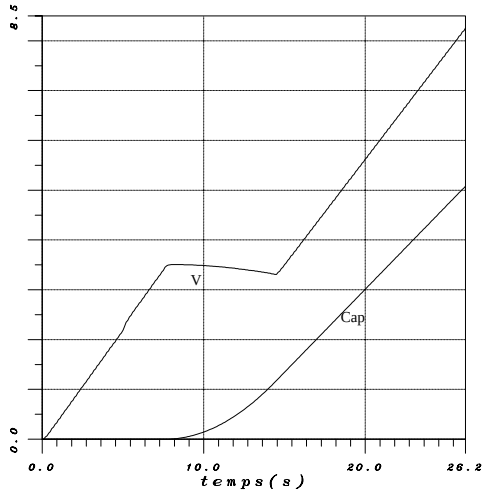
En dérivant n, on obtient une estimation de $\dot{\varphi}$. Bien sûr, plus la période T_e est petite, ou plus N est grand, plus l'estimation de $\dot{\varphi}$ est bonne.



(a) La trajectoire de la voiture.

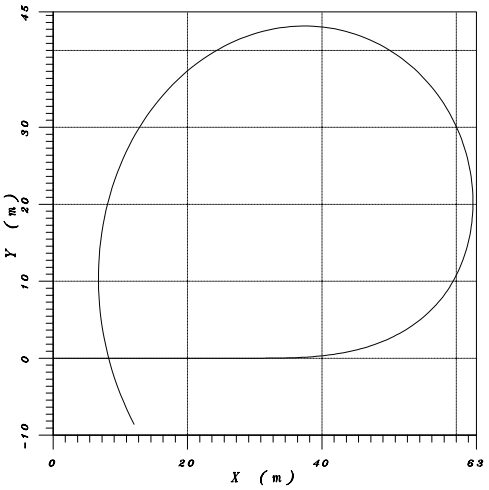


(b) La vitesse réelle et le cap réel du véhicule.

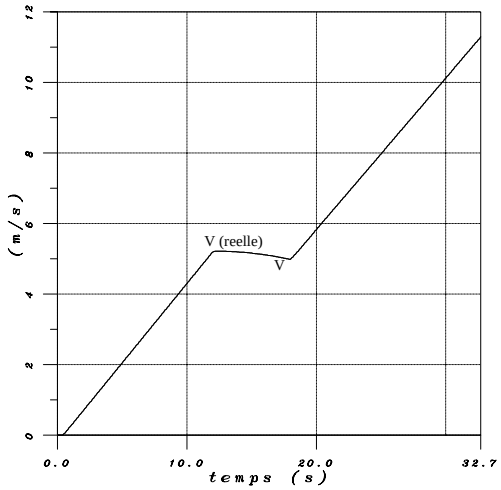


(c) La vitesse et cap par odométrie.

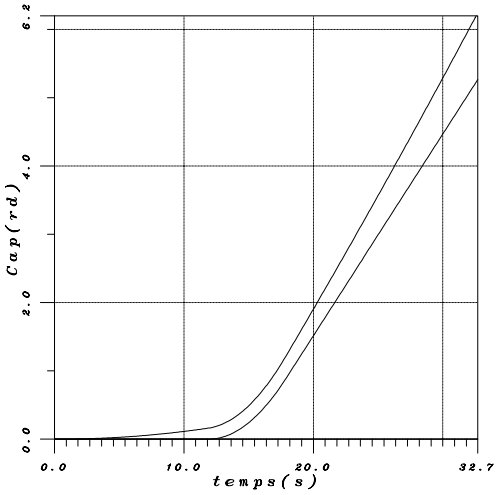
FIG. 2.2-6 – *Simulation avec* $r_3 = r_4$.



(a) Le Traject donné à la voiture.



(b) Les vitesses rélle et estimée par odométrie.



(c) Les caps réel et estimé par odométrie.

FIG. 2.2-7 – *Simulation avec* $dr = 1mm$

Les deux codeurs permettrons de déterminer la position et le cap de la voiture (X, Y, θ) par rapport à sa position initiale. Soient dS et $d\theta$ les déplacements élémentaires de la voiture. Exprimons les en fonction des mesures odométriques des roues n_3 et n_4 pendant un temps T_e, à partir des équations (2.2-20) :

$$dS = \frac{2\pi}{N}\left(\frac{n_3+n_4}{2}r_m + \frac{n_3-n_4}{2}dr \quad\right); \quad d\theta = \frac{2\pi}{N}\left(\frac{n_3-n_4}{(l_1+d)}r_m + \frac{n_3+n_4}{(l_1+d)}dr\right) \tag{2.2-21}$$

Les équations (2.2-21) permettent de déterminer :

- ❏ Le cap θ à l'instant $k+1$: $\theta_{k+1} = \theta_k + d\theta$
- ❏ L'abscisse curviligne S à l'instant $k+1$: $S_{k+1} = S_k + dS$
- ❏ Par contre, les coordonnées X et Y à l'instant $k+1$ sont indéterminées. Le véhicule peut suivre n'importe quelle trajectoire de longueur dS en tournant de $d\theta$. Il faut faire des hypothèses sur la trajectoire suivie entre deux mesures successives. Plusieurs hypothèses sont possibles :

① le véhicule se déplace rectilignement de dS dans la direction de θ_k puis tourne de $d\theta$;

② le véhicule tourne de $d\theta$ puis avance de dS dans la direction θ_k ;

③ le véhicule tourne de $\frac{d\theta}{2}$, avance de dS dans la direction $\theta_k + \frac{d\theta}{2}$ et tourne de $\frac{d\theta}{2}$. Cette dernière hypothèse est la plus souvent retenue. Elle conduit aux équations de récurrences suivantes :

$$X_{k+1} = X_k + dS\cos(\theta_k + \frac{d\theta}{2}) \quad ; \quad Y_{k+1} = Y_k + dS\sin(\theta_k + \frac{d\theta}{2}) \tag{2.2-22}$$

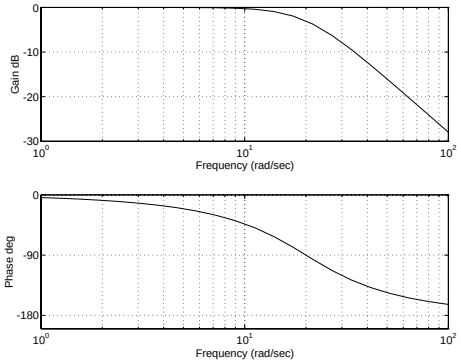
Remarque 2.2.2 *Le modèle reste valable lorsque les roues arrières ne glissent pas. Des améliorations peuvent être faites en intégrant des mesures inertielles avec des gyromètres ou des accéléromètres.*

Estimation de la vitesse de la voiture et de sa dérivée Nous voulons faire une estimation de la vitesse et de l'accélération de la voiture à partir du déplacement élémentaire dS. Appliquons l'algorithme d'estimation, détaillé en Annexe B, sur le vecteur : $\begin{bmatrix} \hat{V}_{odo} & \hat{A}_{odo} \end{bmatrix}^T$ où $\hat{V}_{odo}$ et $\hat{A}_{odo}$ représente respectivement l'estimée de la vitesse et la dérivée de la vitesse estimée. (voir L'algorithme 1)

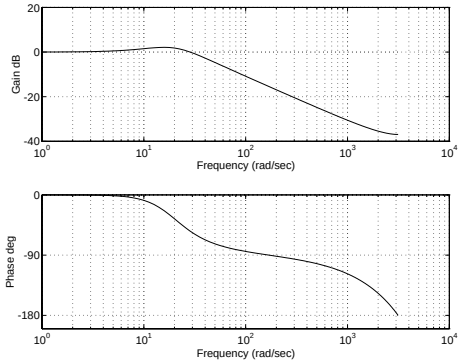
Estimation de la dérivée première et seconde du cap Nous considérons le déplacement élémentaire $d\theta$ comme la mesure de l'estimateur. L'algorithme d'estimation du vecteur $\begin{bmatrix} \hat{\theta}' & \hat{\theta}'' \end{bmatrix}^T$ est détaillé dans l'algorithme 2. $\hat{\theta}'$ est l'estimée de la dérivée du cap de la voiture et $\hat{\theta}''$ sa dérivée seconde.

$$
\begin{aligned}
\hat{V}_{odo,k+1/k} &= \hat{V}_{odo,k/k} + \hat{A}_{odo,k+1/k} T_e \\
\hat{A}_{odo,k+1/k} &= \hat{A}_{odo,k/k} \\
V_{odo,innov} &= (\frac{ds}{T_e} - \hat{V}_{odo,k+1/k}) \\
\hat{V}_{odo,k+1/k+1} &= \hat{V}_{odo,k+1/k} + K^d_{v,odo} V_{odo,innov} \\
\hat{A}_{odo,k+1/k+1} &= \hat{A}_{odo,k+1/k} + K^d_{a,odo} V_{odo,innov}
\end{aligned}
$$

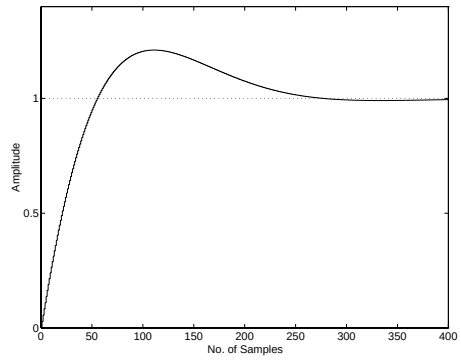
Algorithmme 1: Filtrage du déplacement dS, pour l'estimation de $\hat{V}_{odo}$ et de $\hat{A}_{odo}$.



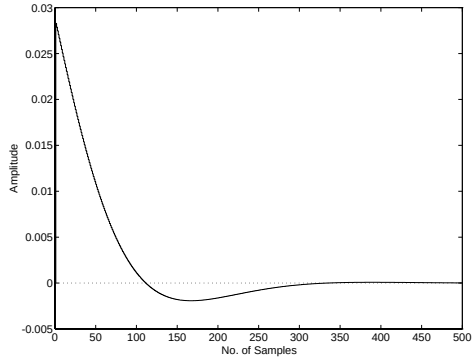
(a) Filtre continu ($\xi = 0.7071$ et $\omega = 20$).



(b) Filtre numérique équivalent.



(c) Réponse indicielle du filtre numérique.

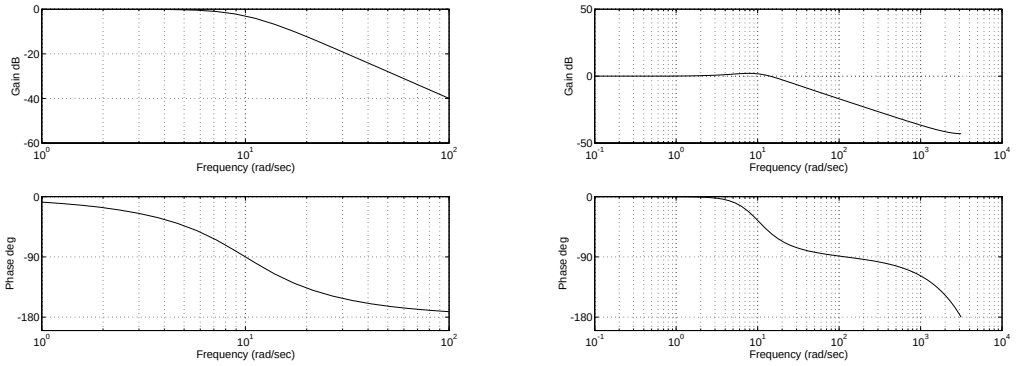


(d) Réponse impulsionnelle du filtre numérique.

FIG. 2.2-8 – *Réglage des gains pour l'estimation de la vitesse et de l'accélération de la voiture à partir de la mesure de* ds *(*$T_e = 1ms$, $K^d_{v,odo} = 0.0283$, $K^d_{a,odo} = 0.3944$*).*

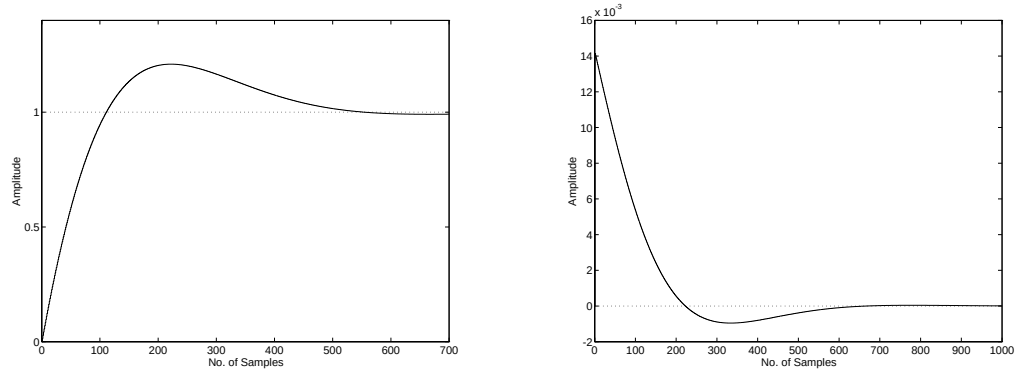
$$
\begin{aligned}
\hat{\theta}'_{k+1/k} &= \hat{\theta}'_{k/k} + \hat{\theta}''_{k+1/k} T_e \\
\hat{\theta}''_{k+1/k} &= \hat{\theta}''_{k/k} \hat{\theta}'_{innov} = (\frac{d\theta}{T_e} - \hat{\theta}'_{k+1/k}) \\
\hat{\theta}'_{k+1/k+1} &= \hat{\theta}'_{k+1/k} + K^d_{dpsi} \hat{\theta}'_{innov} \\
\hat{\theta}''_{k+1/k+1} &= \hat{\theta}''_{k+1/k} + K^d_{d2psi} \hat{\theta}'_{innov}
\end{aligned}
$$

Algorithmme 2: Filtrage du déplacement $d\theta$, pour l'estimation de $\hat{\theta}'$ et de $\hat{\theta}''$.



(a) Filtre continu ($\xi = 0.7071$ et $\omega = 10$).

(b) Filtre numérique équivalent.



(c) Réponse indicielle.

(d) Réponse impulsionnelle.

FIG. 2.2-9 – *Réglage des gains pour l'estimation $\hat{\theta}'$ et $\hat{\theta}''$ à partir de le mesure $d\theta$ ($T_e = 1ms$, $K^d_{dpsi,odo} = 0.0141$, $K^d_{d2psi,odo} = 0.0993$).*

2.2.6 Modèle de la direction

Physiquement, les orientations des roues de direction 1 et 2 sont réalisées à l'aide d'un différentiel de direction. Ce différentiel transmet mécaniquement l'angle β_v donné à la roue virtuelle avant aux deux roues avec les angles β_1 et β_2. La transmission est conçue de telle sorte que l'emplacement du centre instantané de rotation (CIR) soit situé à l'intersection des perpendiculaires aux deux roues avant issues de leur centre ainsi que la droite matérialisant le train de roues arrières.

Les équations 3 et 4 de (2.2-15) ($\left(V^{r_3}_{C_3/O}\right)_{\vec{j}_r} = \left(V^{r_4}_{C_4/O}\right)_{\vec{j}_r} = 0$) expriment la même contrainte de non glissement latéral des roues 3 et 4. Les vitesses latérales exprimées par les équations 1 et 2 de (2.2-15) des deux roues avants ($\left(V^{r_1}_{C_1/O}\right)_{\vec{j}_r} = 0$ et $\left(V^{r_2}_{C_2/O}\right)_{\vec{j}_r} = 0$) sont dépendantes et peuvent être reconstituées à partir de la vitesse latérale au point de contact de la roue virtuelle avant et celle d'une des roues arrières.

Le lien entre β_v et β_1 s'obtient, en écrivant que les vecteurs $V^{r_1}_{C_1/O,x}$, $V^{r_{va}}_{C_{va}/O,x}$ et $V^{r_{vr}}_{C_{vr}/O,x}$sont dépendants :

$$\det \begin{bmatrix} -\sin\beta_1 & \cos\beta_1 & -l_1\sin\beta_1 + l_2\cos\beta_1 \\ \sin\beta_v & -\cos\beta_v & -l_2\cos\beta_v \\ 0 & 1 & 0 \end{bmatrix} = 0$$

$$\beta_1 = \arctan\frac{l_2\sin\beta_v}{l_1\sin\beta_v + l_2\cos\beta_v} \tag{2.2-23}$$

De même pour β_v et β_2 :

$$\det \begin{bmatrix} -\sin\beta_2 & \cos\beta_2 & l_1\sin\beta_2 - l_2\cos\beta_2 \\ \sin\beta_v & -\cos\beta_v & -l_2\cos\beta_v \\ 0 & 1 & 0 \end{bmatrix} = 0$$

$$\beta_2 = \arctan\frac{l_2\sin\beta_v}{l_1\sin\beta_v - l_2\cos\beta_v} \tag{2.2-24}$$

Estimation de la vitesse du volant L'estimation de la vitesse du volant $\hat{v}$ à partir de la mesure de sa position α peut se faire en appliquant le filtre (Annexe B) sur le vecteur suivant : $\begin{bmatrix} \hat{\alpha} & \hat{v} \end{bmatrix}^T$

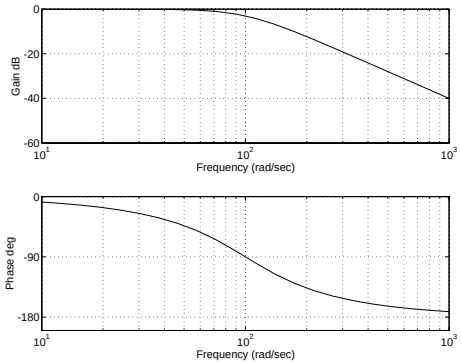
Si on suppose que la position du volant α est obtenue directement à partir de la mesure des tops, n_v, du codeur du volant : $\alpha = An_v + B$ avec A la constante de conversion du nombre de tops en radians, et B la constante qui permet de centrer le volant alors l'estimation de la vitesse peut être obtenue par l'algorithme 3.

2.2.7 Modèle cinématique de locomotion

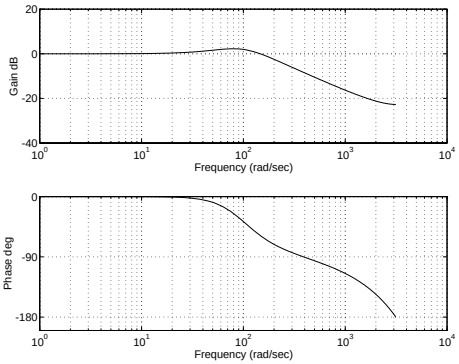
La chaîne de traction comprend le moteur électrique au milieu de l'essieu avant. Le moteur transmet sa vitesse angulaire $\dot{\varphi}_m$ aux deux roues avant 1 et 2, grâce à un différentiel mécanique. Les relations entre les vitesses de rotations $\dot{\varphi}_m$, $\dot{\varphi}_1$ et $\dot{\varphi}_2$ peuvent être obtenues en écrivant que les équations de non glissement longitudinale (équation 2.2-14) sont linéairement dépendants.

$$
\begin{aligned}
\hat{\alpha}_{k+1/k} &= \hat{\alpha}_{k/k} + \hat{v}_{k+1/k} T_e \\
\alpha_{innov} &= (\alpha - \hat{\alpha}_{k+1/k}) \\
\hat{\alpha}_{k+1/k+1} &= \hat{\alpha}_{k+1/k} + K^d_{v,p} \alpha_{innov} \\
\hat{v}_{k+1/k+1} &= \hat{v}_{k+1/k} + K^d_{v,v} \alpha_{innov}
\end{aligned}
$$

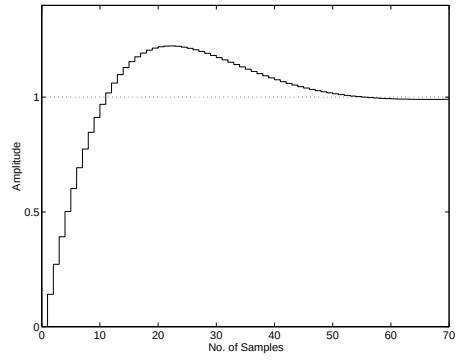
Algorithmme 3: Estimation de l'angle $\hat{\alpha}$, et de la vitesse $\hat{v}$, du volant.



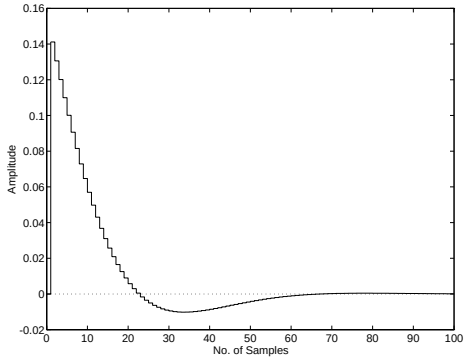
(a) Filtre continu ($\xi = 0.7071$ et $\omega = 100$).



(b) Filtre numérique équivalent.



(c) Réponse indicielle.



(d) Réponse impulsionnelle.

FIG. 2.2-10 – *Réglage des gains pour l'estimation de la vitesse du volant à partir de la mesure du "codeur volant" ($T_e = 1ms$, $K^d_{v,p} = 0.1412$, $K^d_{v,v} = 9.3137$).*

Capteur	Mesure : y	Estimation : $\hat{x}^T$	$remarques$
volant n_v	$\alpha = An_v + B$	$\begin{bmatrix} \hat{\alpha} & \hat{v} \end{bmatrix}$	
moteur n_m	$V_m = Cn_m$	$\begin{bmatrix} \hat{V}_m & \hat{A}_m \end{bmatrix}$	
roue gauche n_4 roue droite n_3	$dS = \frac{n_3+n_4}{2} r_m$ $+\frac{n_3-n_4}{2} dr$ $S_{k+1} = S_k + dS$	$\begin{bmatrix} \hat{V}_{odo} & \hat{A}_{odo} \end{bmatrix}$	$X_{k+1} = X_k$ $+dS\cos(\theta_k + \frac{d\theta}{2})$ $Y_{k+1} = Y_k$ $+dS\sin(\theta_k + \frac{d\theta}{2})$
roue gauche n_4 roue droite n_3	$d\theta = \frac{n_d-n_4}{2} r_m$ $+\frac{n_d+n_4}{2} dr$ $\theta_{k+1} = \theta_k + d\theta$	$\begin{bmatrix} \hat{\theta\prime} & \hat{\theta\prime\prime} \end{bmatrix}$	

TAB. 2.2-1 – *Les Mesures-Estimations des capteurs proprioceptifs.*

Estimation de la vitesse du moteur et de sa dérivée Soit V_m la vitesse du moteur, obtenue directement à partir de la mesure des tops du codeur moteur : $V_m = Cn_m$. C est la constante de conversion du nombre de tops en mètres. Appliquons l'algorithme de filtrage (voir annexe) sur le vecteur $\begin{bmatrix} \hat{V}_m & \hat{A}_m \end{bmatrix}^T$ où $\hat{A}_m$ est l'estimée de la dérivée de la vitesse.

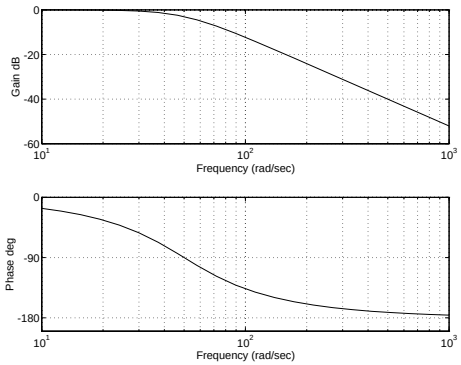
$$
\begin{aligned}
\hat{V}_{m,k+1/k} &= \hat{V}_{m,k/k} + \hat{A}_{m,k+1/k} T_e \\
V_{m,innov} &= (V_m - \hat{V}_{m,k+1/k}) \\
\hat{V}_{m,k+1/k+1} &= \hat{V}_{m,k+1/k} + K^d_{m,v} V_{m,innov} \\
\hat{A}_{m,k+1/k+1} &= \hat{A}_{m,k+1/k} + K^d_{m,a} V_{m,innov}
\end{aligned}
$$

Algorithmme 4: Estimation de la vitesse $\hat{V}_m$ du moteur et de sa dérivée $\hat{A}_m$.

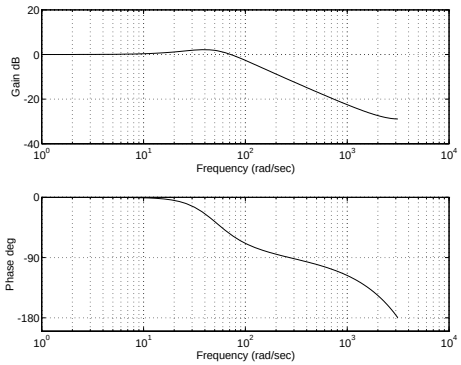
2.2.8 Fonction de transfert du moteur de locomotion

La traction de la voiture, équivalente à une force appliquée, dépend des caractéristiques du moteur de locomotion. Nous avons procédé à des essais pour identifier le moteur. La figure 2.2-12 représente la fonction de transfert de la force motrice de la voiture. On voit que la courbe est formée par deux groupes représentant les deux régimes du moteur avec une continuité aux points critiques. La tension U est exprimée en pourcentage de la valeur maximale $U_{\max}$ fournie par le convertisseur Numérique/Analogique. Cela permettra d'avoir une fonction de transfert indépendante du convertisseur. Si l'on appelle ddp la tension aux bornes du C.N.A : $ddp = U_{\max} U$

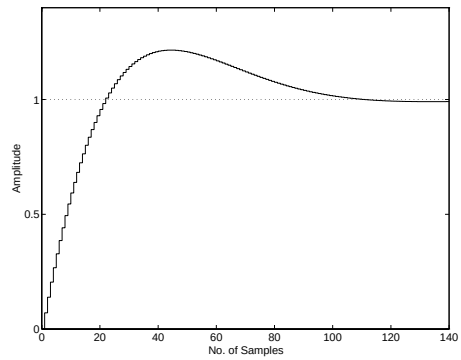
En inversant la caractéristique de la fonction $F = f(U, V)$, on obtient les formules donnant la tension U en fonction de la force motrice F et de la vitesse V (Algorithme 5).



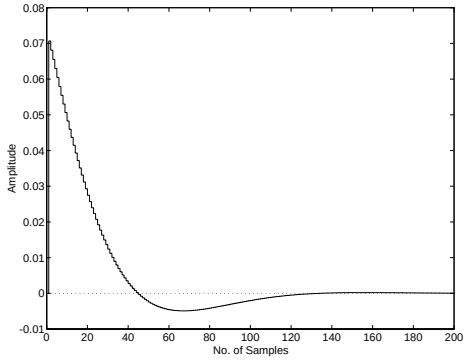
(a) Filtre continu ($\xi = 0.7071$ et $\omega = 50$).



(b) Filtre numérique équivalent.



(c) Réponse indicielle.



(d) Réponse impulsionnelle.

FIG. 2.2-11 – *Réglage des gains pour l'estimation de la vitesse et l'accélération du moteur à partir de la mesure du "codeur moteur" ($T_e = 1ms$, $K^d_{m,v} = 0.0707$, $K^d_{m,a} = 2.4132$).*

En accélération :

$$U = \begin{cases} U_1 = \frac{F}{(1720-10V)+0,12} & \text{si } U \leq 0,45 \\ U_2 = \frac{F}{(2800-40V-3,5V^2)+0,25} & \text{si } U \geq 0,55 \\ 10(U_2(U_1-0,45)+U_2(0,55-U_1)) & \text{si non} \end{cases}$$

En décélération :

$$U = \begin{cases} U_1 = \frac{F}{(1900+35V)-0,08} & \text{si } U \leq -0,3 \\ U_2 = \frac{F}{(2400+50V)-0,14} & \text{si } U \geq -0,4 \\ 10(U_1(U_1+0,4)-U_2(0,3+U_1)) & \text{si non} \end{cases}$$

avec :
U_1, U_2 et U_3 exprime le pourcentage de la tension appliquée ;
V la vitesse de la voiture en $m.\sec^{-1}$;
F la force motrice en Newton.

Algorithmme 5: Formule d'inversion de la caractéristique du moteur.

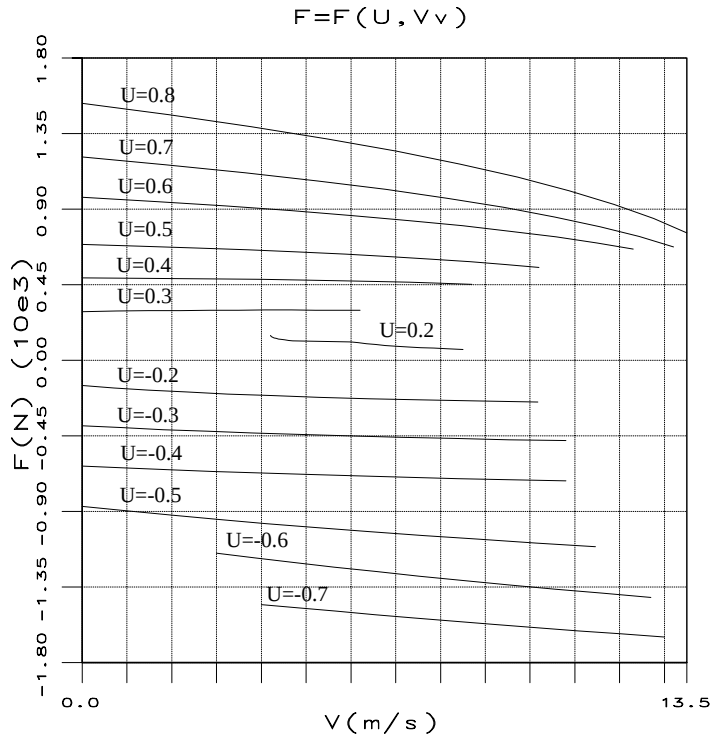


FIG. 2.2-12 – *Force électrique du moteur.*

Remarque 2.2.3 *Si la vitesse est positive, la demande d'une consigne U négative permet de freiner. Deux avantages découlent de cette procédure : la récupération de l'énergie cinétique de la voiture en rechargant les batteries, et le freinage sans blocage des roues (ce qui limite les glissements).*

2.3 Les capteurs extéroceptifs

Les capteurs extéroceptifs réalisent leurs mesures par rapport à l'environnement extérieur. Ils diffèrent des capteurs proprioceptifs, comme par exemple les odomètres, qui analysent la position et le mouvement du robot indépendamment de l'environnement. Pour les besoins de nos applications de suivi et de parking, nous avons élaboré deux capteurs :

- Le premier, basé sur la vision permet de localiser la position relative d'un véhicule par rapport à un autre [PDA95] ;
- Le deuxième utilise les capteurs à ultrasons pour faire une reconstruction de l'environnement [DAP96].

2.3.1 Le capteur de positions

Le capteur de position est disposé sur le véhicule comme illustré sur la figure 2.3-13. Une cible de diodes infrarouges est disposée en arrière et une caméra est placée au milieu de l'essieu avant, un repère $R_c(O_c, \vec{i}_c, \vec{j}_c)$ lui est associé. La cible est placée au milieu de l'essieu arrière, son repère est confondu avec le repère de la voiture R_1.

Modèle de la caméra

La caméra linéaire (associée à une optique cylindrique) permet de transformer un point 2D en un point 1D. Soit un point P du plan ayant pour coordonnées (x_c, y_c) dans R_c. Si ce point est dans le champ de la caméra, alors l'abscisse de ce point sur la barrette, ayant pour repère $R_b(O_b, \vec{i}_b)$, est exprimée par :

$$si\ \left(d_{c\,\min} < x_c < d_{c\,\max}\ et\ \arctan\left(\frac{y_c}{x_c}\right) \leq \alpha_{c\,\max}\right)\ alors\ x_b = f\frac{y_c}{x_c}$$

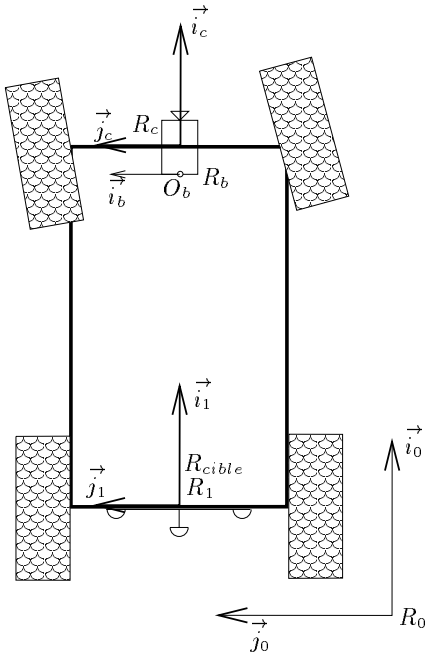
où : $d_{c\,\min}$ (resp. $d_{c\,\max}$), $\alpha_{c\,\max}$ et f représentent respectivement la distance minimale (resp. maximale) de vision, l'angle maximal de vision et f la distance focale de la caméra.

Calcul de positionnement

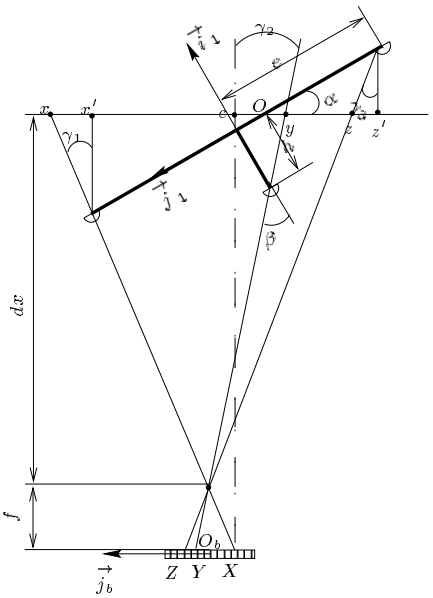
Connaissant la géométrie de la cible (e, h), la distance focale f et les coordonnées (X, Y, Z) des trois diodes infrarouges dans le repère R_b, on veut déduire : la distance dx, la déportation latérale en distance dy, et l'orientation $d\theta$ entre la cible et la caméra.

posons : $t = \tan(d\theta)$ et $dc = \frac{dx}{\cos(d\theta)}$

Par des relations géométriques [Dav93], on trouve :



(a) Disposition cible-caméra sur la voiture.



(b) Projection sur la barrette de la caméra.

FIG. 2.3-13 – *Le capteur de position.*

$$t = \frac{2(X+Z-2Y)ef + 2Yh(X-Z)}{e(X-Z)^2 + 2fh(X-Z) - e(X+Z-2Y)(X+Z)} \tag{2.3-25}$$

$$dc = \frac{ef}{X-Z}\left(2 + t\frac{X+Z}{f}\right) \tag{2.3-26}$$

Une fois ce calcul fait, on déduit :

$$\begin{aligned} d\theta &= \arctan(t) \\ dx &= dc\cos(d\theta) \\ dy &= \tfrac{dx}{f}Y + h\cos(d\theta)\left(t - \tfrac{Y}{f}\right) \end{aligned} \tag{2.3-27}$$

Calculs d'erreur On cherche à calculer l'erreur sur la mesure $d = X, Y, Z$, à partir d'une tache sur la barrette de la caméra, et en fonction de la distance, dx, entre la caméra et la cible. Si on se place dans le cas où l'angle entre la cible et la caméra est faible, $d\theta \approx 0$, alors :

$$d = \frac{ef}{d}x \Rightarrow \Delta d = \frac{d^2}{ef}\Delta(dx) \tag{2.3-28}$$

Détermination de la vitesse relative

La vitesse relative entre la cible et la caméra (c'est à dire entre les deux véhicules qui se suivent) est estimée à partir des informations de positionnement. Nous avons utilisé le filtre de *Kalman*.

Considérons le vecteur d'état suivant : $X = \begin{pmatrix} da & dv & dx \end{pmatrix}^T$. Le modèle est : $\dot{X} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} X + \begin{pmatrix} w \\ 0 \\ 0 \end{pmatrix}$ et la mesure : $Y = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} X + v$

où v et w représentent respectivement le bruit du modèle et le bruit de mesure. Ils sont supposés indépendants, blancs, et suivent des lois normales de probabilité avec des variances respectives Q et R.

$$p(w) \sim N(0, Q) \; et \; p(v) \sim N(0, R)$$

Rappel sur le Filtrage de Kalman discret

$$X_{k+1} = A_k X_k + w_k = \begin{pmatrix} 1 & 0 & 0 \\ T_e & 1 & 0 \\ T_e^2/2 & T_e & 1 \end{pmatrix} X_k + \begin{pmatrix} w_a \\ 0 \\ 0 \end{pmatrix}$$

$$Y_k = C_k X_k + v_k = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} X_k + v_x$$

Le filtre de Kalman est basé sur un processus de rétro-action (figure 2.3-14) : le filtre estime l'état du système et la rétro-action est faite à partir des mesures. Ce qui fait que l'algorithme est composé de deux étapes : l'étape de *prédiction (d'entretien ou de propagation)* et l'étape d'*estimation (de mise à jour ou de recalage)*.

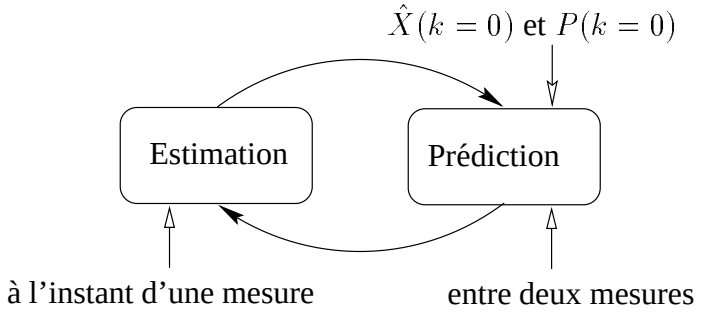


FIG. 2.3-14 – *Le cycle dans le filtre de Kalman.*

Initialisation du filtre Comme l'état initial est extrêmement incertain, alors on prend un état initial "vraisemblable", et on lui associe une matrice $P(0)$ très grande (à la limite $P^{-1}(0) = 0$)

$$\hat{X}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \; et \; P = \begin{pmatrix} P_{\max} & 0 & 0 \\ 0 & P_{\max} & 0 \\ 0 & 0 & P_{\max} \end{pmatrix}$$

Les équations de prédiction :
(1) Estimation de l'état : $\hat{X}_{k+1/k} = A_k \hat{X}_k$
(2) Estimation de la covariance de l'erreur : $P_{k+1/k} = A_k P_{k/k} A_k^T + Q_k$
Les équations d'estimation
(1) Calcul du gain de Kalman : $K_k = P_{k+1/k} C_k^T (C_k P_{k+1/k} C_k^T + R_k)^{-1}$
(2) Mise à jour de l'estimation avec la mesure Y_k : $\hat{X}_{k+1/k+1} = \hat{X}_{k+1/k} + K(Y_k - C_k \hat{X}_{k+1/k})$
(3) Mise à jour de la covariance de l'erreur : $P_{k+1/k+1} = (I - KC_k) P_{k+1/k}$

Algorithmme 6: Le filtre de Kalman.

Variance du bruit de modèle Le bruit sur le modèle peut être changé continuellement dans le filtrage pour ajuster les dynamiques. L'amplitude de Q_a est réduite quant la cible se déplace lentement par rapport à la caméra, et augmentée lorsque les dynamiques changent rapidement. Dans ce cas, Q_a représente non seulement les incertitudes du modèle, mais aussi les incertitudes dans le mouvement de la cible.

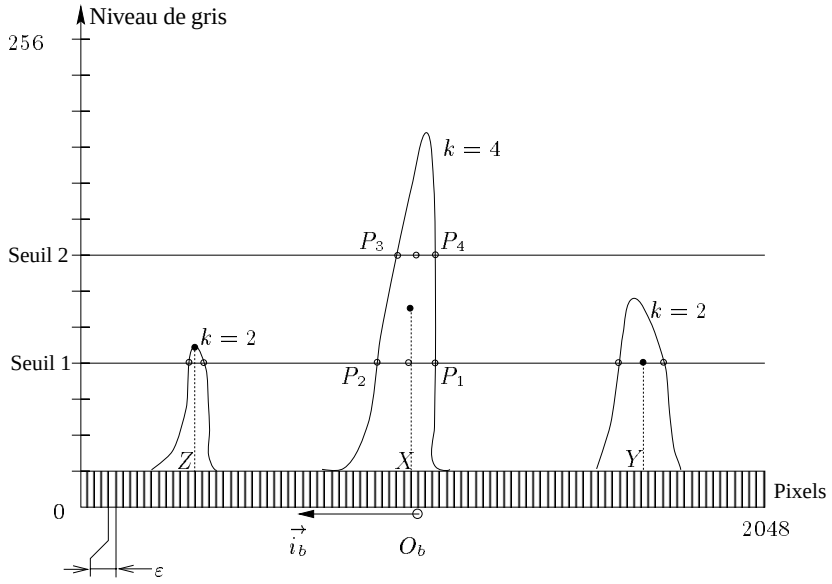


FIG. 2.3-15 – *L'erreur de mesure.*

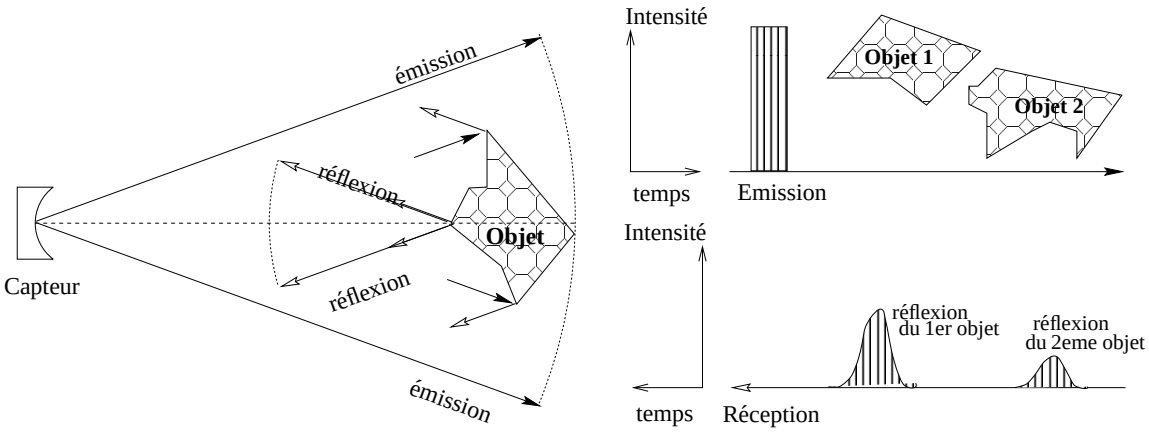
Variance du bruit de mesure Physiquement, la distance d de l'expression (2.3-28) est une distance quantifiée. Elle est déduite de la mesure du nombre N_p de pixels dans la barrette de la caméra (figure 2.3-15). Si ε représente la largeur d'un pixel alors : $d = \varepsilon N_p$. Cette quantification est linéaire. Le maximum de l'erreur de quantification (ou incertitude numérique) est égal au demi-pas de quantification $\epsilon/2$. Il en résulte un bruit de mesure, dont la variance est de : $\frac{1}{12}\left(\frac{\varepsilon^2 N_p^2}{4ef}\Delta(dx)\right)^2$ (formule de *Sheppard*) [Miq85].

Dans notre cas, la mesure des positions des taches (X, Y, Z) n'est pas directe. Elle est effectuée sur une moyenne de deux ou quatre points. Pour $k = \{2, 4\}$, la variance de

l'erreur de mesure sera égale à :

$$R_x = \frac{1}{12k}\left(\frac{\varepsilon^2 N_p^2}{4ef}\Delta\left(d_x\right)\right)^2$$

2.3.2 Les capteurs à ultrasons



(a) Cône d'émission et de réflexion.

(b) Emission du capteur et réflexion des objets.

FIG. 2.3-16 – *Fonctionnement d'un capteur à ultrsons.*

Les capteurs à ultrasons appartiennent à la famille des capteurs télémétriques. La distance est obtenue à partir de la mesure du temps de vol d'une onde acoustique et connaissant la vitesse de propagation du son dans l'air. Les capteurs à ultrasons possèdent un cône d'émission et un cône de réception. (voir figure 2.3-16(a))

La figure 2.3-16(b) montre le fonctionnement d'un capteur. A partir de l'émission d'un signal, un écho est réfléchi lorsque le signal émis rencontre un objet.

Modèle d'émission-réception

A l'émission, le capteur émet un signal ayant une intensité $I_{capteur}$. Cette intensité diminue en fonction de la distance parcourue suivant la loi : $I_d = I_{capteur}\left(\frac{1}{d}\right)^2$

Quant l'onde émise rencontre un objet il y aura réflexion. Et l'objet va se comporter comme un centre d'émission : $I_{reception} = I_{objet}\left(\frac{1}{d_2}\right)^2$ où : d_2 représente la distance entre le récepteur et l'objet.

Or, l'intensité I_{objet} que l'obstacle réfléchit provient du capteur. Si l'objet est situé à une distance d_1 de l'émetteur du capteur alors : $I_{reception} = I_{capteur}\left(\frac{1}{d_1}\right)^2\left(\frac{1}{d_2}\right)^2$

L'émetteur et le récepteur sont physiquement au même endroit ($d_1 = d_2 = d$), donc l'intensité à la réception est atténuée suivant la formule : $I_{reception} = I_{capteur}\left(\frac{1}{d}\right)^4$

Influence de la température La vitesse de propagation des ondes ultrasonores dépendent de la température suivant la formule :

$$c(T) = 20\sqrt{T} \qquad T \ : température\ en\ ^0K$$

Si le capteur fonctionne dans un environnement où la température varie entre -20^0C et 50^0C alors la vitesse de propagation varie entre $318\ m.s^{-1}$ et $360\ m.s^{-1}$. Si on mesure un temps égale à $t = 60\ ms$, par exemple, ceci entraîne une variation de distance allant de $9.55\ m$ à $10.8\ m$. L'erreur est d'environ $1.25\ m$, il est donc nécessaire de tenir compte de la température.

Influence de l'effet Doppler L'effet apparaît lorsque le "système" équipé des capteurs à ultrasons se déplace. Si la vitesse du "système" à l'émission est v_e(resp. réception v_r), et la fréquence d'émission est f_e, alors la fréquence reçue f_r est :

$$f_r = f_e(\frac{1}{1 \pm \frac{v_e}{c}})(1 \mp \frac{v_r}{c})$$

suivant que le "système" se déplace dans la même direction de l'émission des ultrasons où dans le sens inverse. Par exemple, si le "système" se déplace à $10\ m.s^{-1}$à la température de 20^0C, et le capteur émet à la fréquence de $45\ KHz$, alors la fréquence reçue varie entre $42.4\ KHz$ et $47.7\ KH_z$. Il faut s'assurer que les caractéristiques du capteur permettent de détecter le signal dans cette gamme de fréquence.

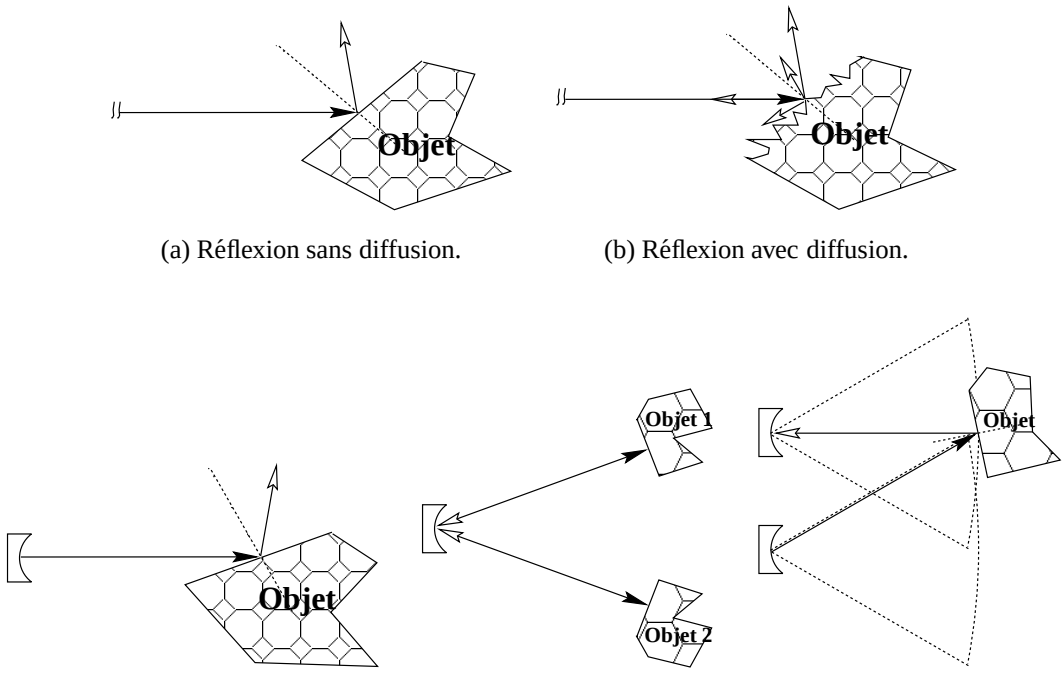
Comportement des capteurs à ultrasons

- Les mesures données par les capteurs à ultrasons dépendent de l'état de surface. La réflexion dépend de l'état de surface de l'objet. Un objet ayant une surface "régulière" (figure 2.3-17(a)) est détecté plus facilement qu'un objet ayant une surface "irrégulière" (figure 2.3-17(b)) ;

- Le cône de réception est relativement étroit (de 10^0 à 20^0). La conséquence directe est que l'on ne peut obtenir de mesure que si la différence d'angle entre la perpendiculaire à la surface réfléchissante et l'axe principal d'émission est inférieure au demi angle d'émission (figure 2.3-17(c)) ;

- Si plusieurs objets sont localisés à la même distance par rapport au capteur (figure 2.3-17(d)), la réflexion donne le même temps de mesure. Il faut donc utiliser plusieurs capteurs pour pouvoir localiser les objets dans un plan ;

- L'utilisation de plusieurs capteurs peut créer une interférence qui fausse la mesure (figure 2.3-17(e)).



(a) Réflexion sans diffusion.

(b) Réflexion avec diffusion.

(c) Réflexion hors champs.

(d) Même distance pour deux objets.

(e) Intéraction entre les capteurs.

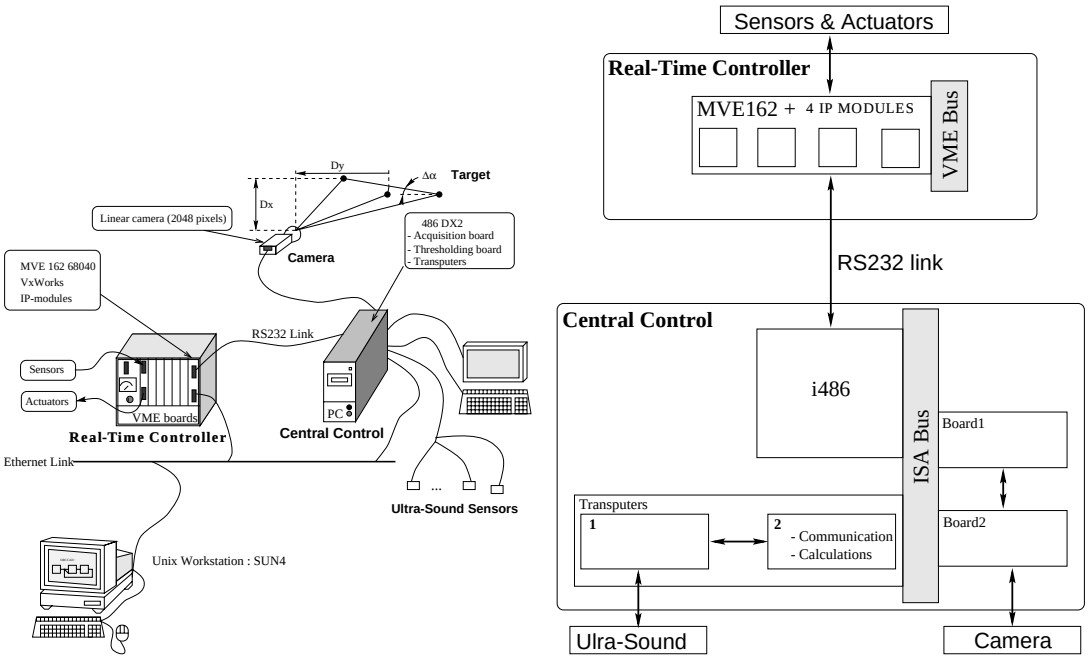
FIG. 2.3-17 – *Comportements des capteurs à ultrasons.*

2.4 Architecture de l'informatique embarquée

L'architecture embarquée est constituée de deux unités (figure 2.4-18(a)) appelées respectivement : l'unité centrale de commande CC et l'unité temps réel de contrôle RTC. Chaque unité peut contenir un ou plusieurs processeurs. Globalement, l'unité CC s'occupe de la gestion des deux capteurs extéroceptifs (caméra et ultrasons), et l'unité RTC contiendra les algorithmes de contrôle temps réel de la voiture. La communication entre les deux unités (PC et MVE162) se fait par la liaison série RS232 fonctionnant à une vitesse de $38.4Kbit.s^{-1}$ (voir figure 2.4-18(b)). La spécification, la vérification et la génération du code temps réel sont faites sur une station de travail (Sun 4) ; le chargement du code se fait par l'intermédiaire du réseau Ethernet.

2.4.1 L'unité temps réel de contrôle (RTC)

La commande de la voiture est gérée avec la carte MVE162 (68040 25Mhz et 4Mo de RAM) sur laquelle sont placées quatre cartes filles IP-modules, par lesquelles sont connectées les capteurs et les actionneurs de la voiture. Les cartes filles sont disposées directement sur la carte, ce qui fait que la communication se fait directement par mémoire partagée.



(a) Les unitées CC et RTC. (b) Communication entre les modules.

FIG. 2.4-18 – *L'architecture informatique embarquée.*

2.4.2 L'unité centrale de commande (CC)

Cette unité sert à gérer :

Le capteur de localisation Ce capteur est constitué d'une caméra linéaire de 2048 pixels et d'un PC 486-66Mhz avec deux cartes de traitements d'image (iSM307 et iSM237). La carte d'acquisition iSM307 travaille à $1KHz$. Le traitement d'image consiste à trouver, à partir de l'image acquise, les émetteurs infrarouges sur la barrette CCD (c'est à dire le triplet (X, Y, Z)). Pour que le traitement puisse suivre la fréquence d'acquisition, il faut utiliser une carte de seuillage iSM237 qui communique directement avec la iSM307 grâce à un Bus local. Le traitement des données (triangulation et le filtrage) est aussi intégré dans cette unité. Les données traitées sont envoyées vers l'unité RTC pour être prises en compte dans d'asservissement de la voiture.

Les capteurs ultrasonores Les capteurs choisis sont ceux de la série Polaroïd 9000, pour leur faible coût et leur étanchiété [Mou95]. Ces capteurs sont livrés avec une carte de commande associée à chacun d'eux et permettant, par le biais de signaux particuliers, de lancer une mesure et de recevoir un signal indiquant la réception d'un écho. L'émission et la réception sont commandées indépendamment, ce qui permet de faire du multi-échos. La figure 2.4-19 montre la disposition de ces capteurs autour de la voiture. Vu le nombre de capteurs, deux Transputers sont utilisés pour pouvoir les traitées. Le premier s'occupe de

l'émission et de l'acquisition des signaux, et le deuxième se charge de communiquer les données vers le processeur 486 du PC.

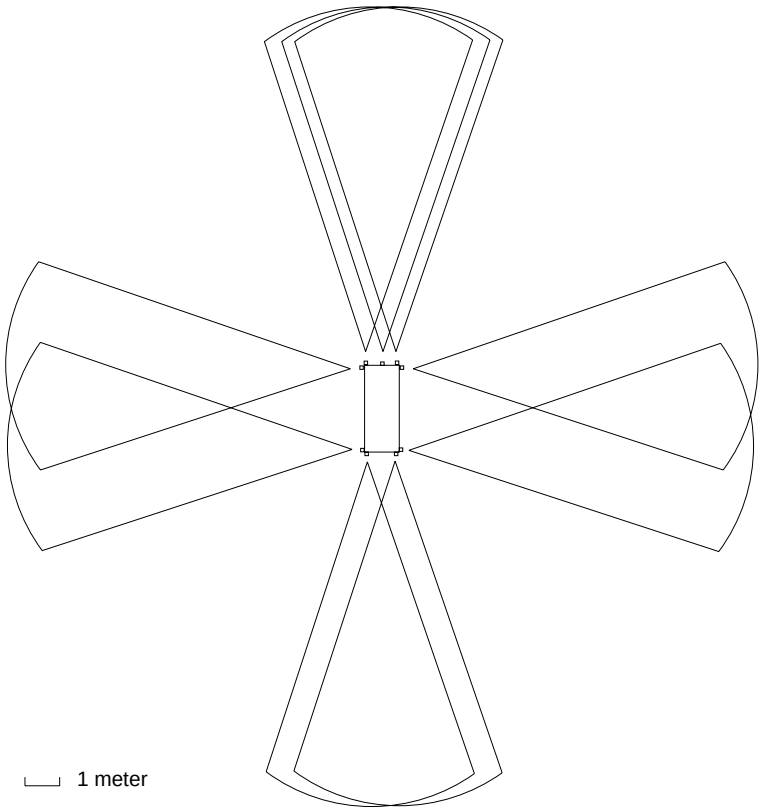


FIG. 2.4-19 – *Disposition des capteurs à ultrasons.*

2.5 Conclusion

Nous avons présenté dans ce chapitre, les fonctionnalités de base permettant le contrôle-commande du prototype ATALANTE. Elles ont été intégrées dans l'environnement ORCCAD suivant la classification proposée dans le chapitre [**?**] grâce à son entité élémentaire : la TM.

Pour mettre en oeuvre le traitement des données capteurs proprioceptifs, nous avons choisi une méthode générique de modélisation des robots mobiles à roues [CBAN96]. A travers cette méthode, nous avons présenté en détail, comment à partir d'un robot mobile à roues et d'un ensemble de capteurs proprioceptifs nous pouvons estimer les grandeurs utiles dans l'établissement des lois de commande. Le modèle utilisé n'est valable que si les roues ne glissement pas. Des travaux sur le prototype ATALANTE ont été entamés pour tenir compte des faibles glissements causés par les déformations des pneus en utilisant une centrale inertielle [Ler98].

Nous avons aussi détaillé l'architecture de l'informatique embarquée et les capteurs extéroceptifs. Ces derniers ont été développés pour trouver des solutions à la version "moyen terme" de PRAXITÈLE définit par les thèmes 8 et 9 : c'est à dire la conduite en pelotons et

l’automatisation du parking. Ces deux réalisations seront traitées dans la deuxième partie du document.

Chapitre 3

Le Train de Véhicules

UN des problèmes traités dans le programme PRAXITÈLE est le ramassage des véhicules vides pour les redistribuer [AADP96]. Nous avons étudié un mode de déplacement en *train virtuel* (Version 2 du programme), où le véhicule de tête est conduit par un chauffeur professionnel et le reste des véhicules le suit sans accrochage matériel. Le *train* peut avoir une longueur maximale de six véhicules. L'opération peut être décomposée en trois étapes :

❶ accrochage entre un véhicule et le *train* qui passe,

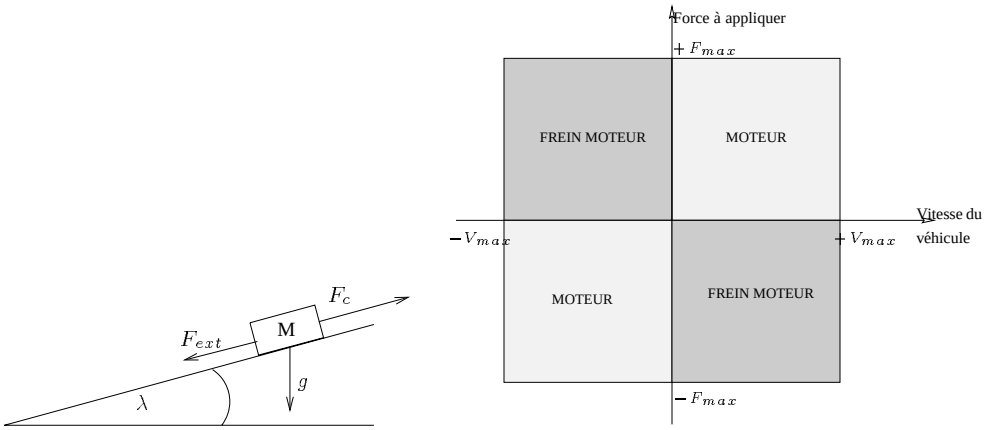
❷ suivi à une vitesse maximale de $50\ km.h^{-1}$ et une inter-distance inférieure à $4\ métres$,

❸ sortie d'un véhicule du *train* pour se garer.

Dans ce chapitre nous allons spécifier, vérifier et implémenter le déplacement du peloton en utilisant ORCCAD et les fonctionnalités réalisées dans le chapitre 2. Dans un premier temps, nous appliquerons le formalisme de TR pour effectuer le contrôle-commande du train virtuel composé de deux véhicules. Puis nous décomposerons cette TR en plusieurs, avant de les recombiner sous forme de PRs. Pour les algorithmes de commande, l'utilisation de l'un ou l'autre des deux formalismes importe peu. Dans le premier cas l'unique TR aura la charge de réaliser les asservissements longitudinale et latérale, alors que dans le deuxième cas plusieurs TRs les partageront. Concernant l'aspect réactif, les PRs manipuleront beaucoup d'événements qui traduisent des situations particulièrement intéressantes à observer. Dans ce cas, la vérification de la cohérence des actions lancées suivant ces situations prend une grande importance. Le plan du chapitre est le suivant :

☞ Nous avons adopté une approche de programmation du bas vers le haut. Dans la section 3.1 nous réaliserons les asservissements des différents moteurs du véhicule en utilisant les fonctionnalités développées précédemment ;

☞ Puis, utilisant le capteur de position développé, et les asservissements décrits dans la section précédente, nous spécifierons la TR qui contrôle le véhicule suiveur ;



(a) Les principales forces. (b) Les modes moteur-frein.

FIG. 3.1-1 – *Modélisation pour le déplacement longitudinal.*

☞ Dans la section 3.3 nous reprogrammerons le suivi sous la forme d'une combinaison de TRs à l'aide du formalisme de PRs.

3.1 Commandes par retour d'état pour l'asservissement des moteurs

Avant de mettre en oeuvre une loi de commande qui fait intervenir les capteurs extéroceptifs, il est nécessaire d'établir les asservissements des différents moteurs du système. Dans cette section nous allons décrire les contrôleurs qui réaliserons les asservissements des différents moteurs du prototype ATALANTE [DP95, DP96a, DP96b]. Pour cela, nous aurons besoin des fonctionnalités donnant l'état interne de la voiture, c'est à dire des TMs de la classe AQ-PR. De nouvelles fonctionnalités appartenant aux classes CA-CO et AC devront être ajoutées pour compléter la librairie des TMs.

3.1.1 Contrôleur du moteur de locomotion

Une modélisation très simple du véhicule est représentée dans la figure 3.1-1(a). Le système est constitué :

– d'une masse M,

– d'un plan faisant un angle λ avec l'horizontal.

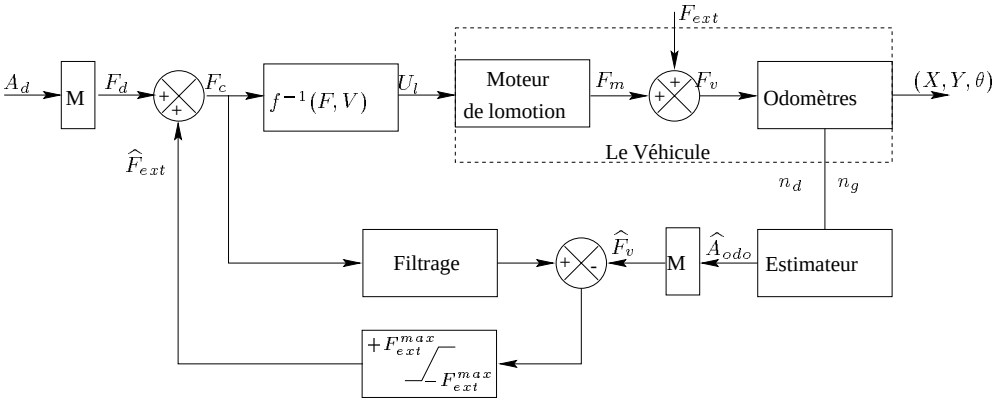
En appliquant la relation fondamentale de la dynamique au système, on aura :

$$F_d = M A_d = F_c - F_{ext}$$

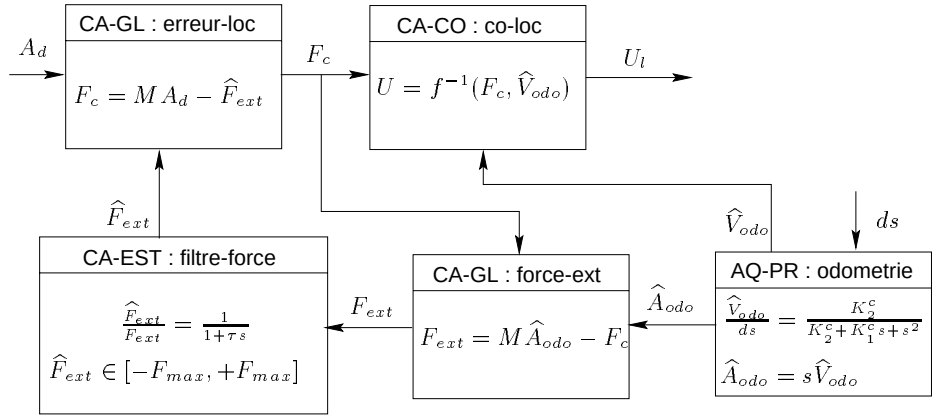
où :

- A_d (resp. F_d) représente l'accélération (resp. la force) que le véhicule désire avoir,
- F_c est la force commandé (par le moteur de locomotion),
- F_{ext} est la force extérieure. Elle est composée par la force de gravité $Mg\sin(\lambda)$ et de la force de freinage F_f.

La consigne, F_c, est calculée à partir d'une référence (la force qu'on désire donner à la voiture MA_d), et d'un estimateur des forces extérieures $\widehat{F}_{ext}$ (voir figure 3.1-2(a)). Comme la réaction du moteur est très rapide, l'estimateur peut être une différence, filtrée et saturée, entre A_d et l'accélération réelle du véhicule. La meilleure estimation de l'accélération du véhicule, $\widehat{A}_{odo}$, est obtenue par l'intermédiaire des capteurs placés sur les roues non motorisées, n_g et n_d. Ainsi, on respecte au mieux l'hypothèse du modèle sans glissement.



(a) Schéma-bloc.



(b) Les TMs de contrôleur de locomotion.

FIG. 3.1-2 – *Contrôleur du moteur de locomotion.*

Caractéristiques du moteur de locomotion Le moteur utilisé pour produire la force F_c est électrique. Il présente l'avantage d'offrir, outre un passage aisé de marche avant à marche arrière, deux modes de fonctionnement (figure 3.1-1(b)) :

– Mode moteur : la puissance du moteur est utilisée comme moteur du mouvement ;

– Mode frein moteur ou récupération : le moteur est alors utilisé comme frein.

La relation qui permet de déterminer la tension à envoyer au C.N.A., connaissant la vitesse de déplacement du véhicule et la force désirée est donnée dans l'algorithme 5 du chapitre 2.

Décomposition du schéma-bloc en TMs Le schéma-bloc de la figure 3.1-2(a), peut être décomposé suivant les TMs de la figure 3.1-2(b) :

① `CA-GL : erreur-loc.` Elle consiste simplement à calculer la consigne en enlevant l'estimé de la force externe $\hat{F}_{ext}$ de la force désirée MA_d.

② `CA-CO : co-loc.` C'est la fonction de transfert du moteur de locomotion détaillée dans le paragraphe 2.2.8 du chapitre 2. Pour une force F_c, cette TM permet de calculer la tension U_l à appliquer au C.N.A du contrôleur du moteur de locomotion en fonction de la vitesse du véhicule $\hat{V}_{odo}$.

③ `AQ-PR : odometrie.` Ce filtre permet d'obtenir une estimation de la vitesse et de l'accélération du véhicule $\left(\hat{V}_{odo}, \hat{A}_{odo}\right)$ à partir des données des odomètres (n_g, n_d). Le détail de ce filtre a été donné dans le chapitre 2.

④ `CA-GL : force-ext.` Cette TM permet d'estimer la force extérieure F_{ext} à partir de la force estimée $M\hat{A}_{odo}$ et de la consigne F_c.

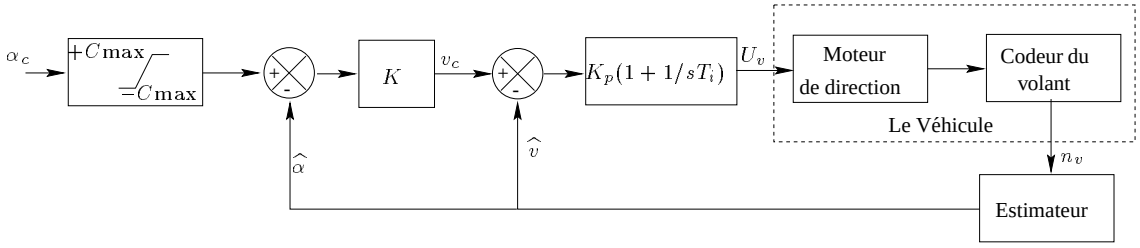
⑤ `CA-EST : filtre-force.` Cette tâche filtre et sature la F_{ext} calculée par `CA-GL` pour obtenir $\hat{F}_{ext}$.

3.1.2 Contrôleur du volant

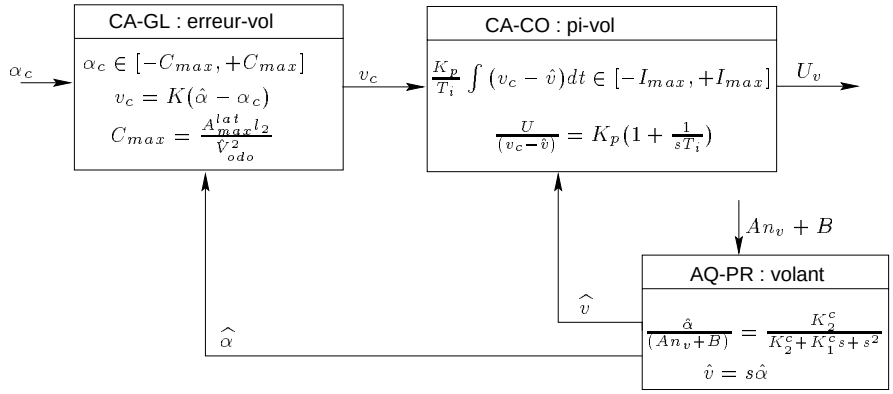
La commande de la direction est faite grâce à un moteur électrique qui agit directement sur le volant. Son inertie est bien moins importante que les frottements pneumatiques (secs). Donc, on a besoin d'un couple fort au début pour commencer à faire tourner les roues mais une fois en mouvement la vitesse maximale est très rapidement atteinte.

Décomposition du schéma-bloc en TMs Le schéma-bloc de la figure 3.1-3(a), peut être décomposé suivant les TMs de la figure 3.1-3(b) :

① `CA-GL : erreur-vol.` Elle consiste simplement à calculer la consigne en vitesse v_c proportionnellement (coefficient K) à l'erreur en position $(\hat{\alpha} - \alpha_c)$. Cette consigne est saturée suivant l'accélération latérale maximale $A^{lat}_{\max}$. On impose que



(a) Schéma-bloc.



(b) Les TMs du contrôleur.

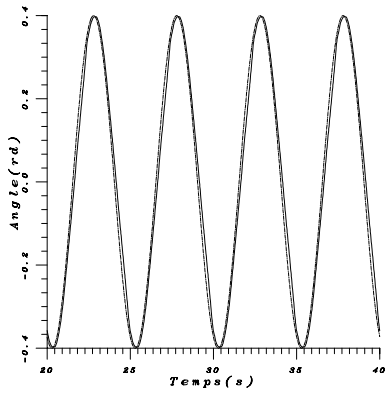
FIG. 3.1-3 – *Le contrôleur du volant.*

α_c soit toujours inférieur à $\frac{A^{lat}_{\max} l_2}{\hat{V}^2_{odo}}$ (l_2 est la distance entre les deux essieux du véhicule, et $\hat{V}^2_{odo}$ est la vitesse du véhicule estimée à partir de l'odométrie) pour ne pas autoriser des consignes énormes lorsque le véhicule roule vite ;

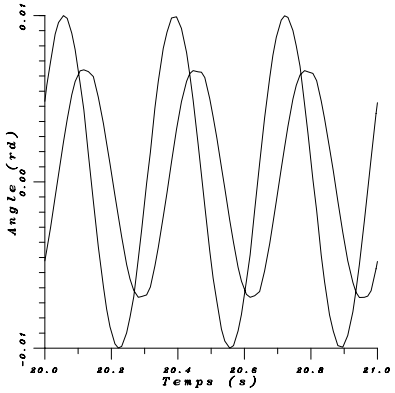
② `CA-CO : pi-vol.` C'est un correcteur de type PI sur la vitesse du volant $\hat{v}$. L'action intégrale $\int \frac{K_p}{T_i} dt$ est saturée entre $\pm I_{i\max}$. K_p est le coefficient proportionnel et T_i est le temps d'intégration de ce correcteur ;

③ `AQ-PR : volant.` Ce filtre permet d'obtenir une estimation de la position et de la vitesse du volant du véhicule $(\hat{v}, \hat{\alpha})$ à partir de la mesure de l'angle du volant n_v. Le détail de ce filtre a été donné dans le chapitre 2.

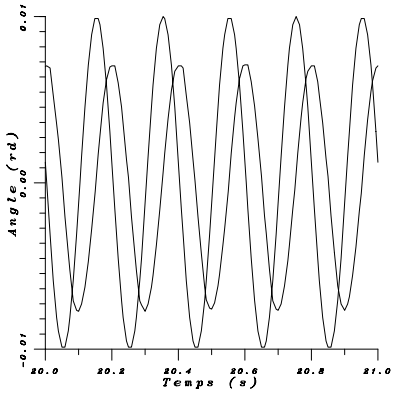
La figure 3.1-4 montre plusieurs expérimentations avec des réglages différents des gains de la loi de commande et des périodes d'activation des TMs. La diminution de la période d'échantillonnage dans la boucle de vitesse permet d'augmenter la fréquence de coupure du contrôleur et d'augmenter les gains.



(a) Réponse à très basse fréquence avec : $K = 15$; $A^{lat}_{\max} = 4\,ms^{-2}$; $K_p = 2$; $T_i = .3\,s$.



(b) Fréquence de coupure est égale à $1.5Hz$ pour : $K = 15$; $A^{lat}_{\max} = 4\,ms^{-2}$; $K_p = 2$; $T_i = .3\,s$ et les périodes des TMs : $T_{erreur-vol} = T_{pi-vol} = T_{volant} = 5\,ms$.



(c) Fréquence de coupure est égale à $5Hz$ pour : $K = 20$; $A^{lat}_{\max} = 4\,ms^{-2}$; $K_p = 4$; $T_i = .04\,s$ et les périodes des TMs : $T_{erreur-vol} = 5\,ms$; $T_{pi-vol} + T_{volant} 1\,ms$.

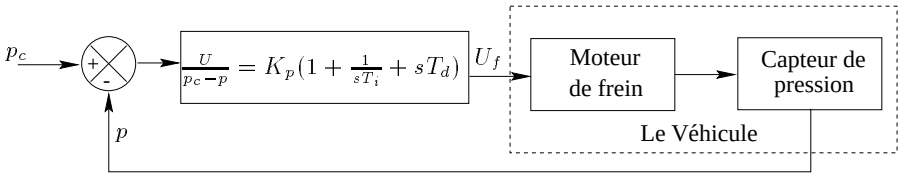
FIG. 3.1-4 – *Influence des périodes d'activation des TMs.*

3.1.3 Contrôleur des freins de route

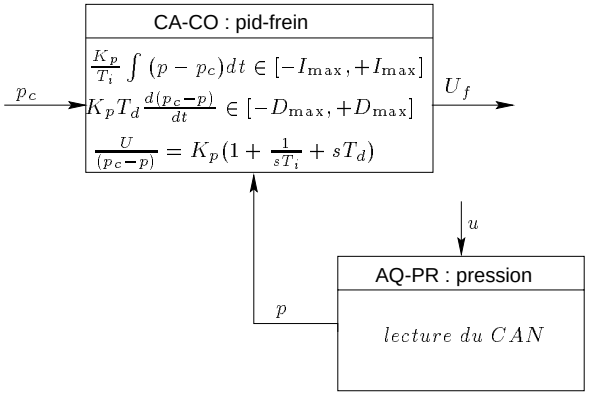
Le freinage est réalisé par un actionneur qui agit sur la pédale. Pour imposer une position de pédale correspondant à une consigne de pression p_c, un étalonnage est effectué :

– "pédale relâchée" correspond à la pression minimale (P_{min}) ;

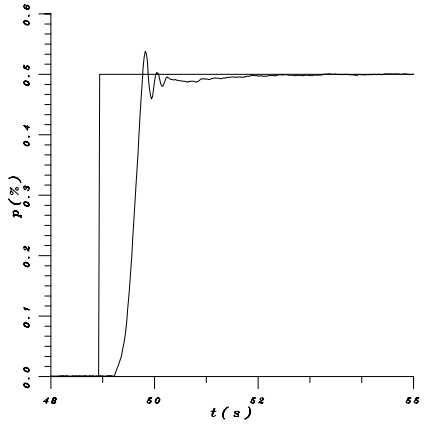
– "pédale à fond" correspond à la pression maximale (P_{max}).



(a) Schéma-bloc du contrôleur des freins de route.



(b) Les TMs du contrôleur.



(c) Réponse à un echelon.

FIG. 3.1-5 – *Contrôleur des freins de route.*

La figure 3.1-5(a) représente le schéma bloc du contrôleur PID réalisé par bouclage sur la pression des freins de route.

Décomposition du schéma-bloc en TMs Vue la simplicité du contrôleur, seulement deux TMs ont été utilisées (figure 3.1-5(b)) :

① `CA-CO : pid-frein.` Cette TM consiste à calculer la consigne à envoyer au C.N.A. en utilisant un correcteur de type PID sur la pression. Les actions intégrale $\frac{K_p}{T_i}\int (p_c - p)$ et dérivée $K_pT_d\frac{(p_c-p)}{dt}$ sont saturées respectivement entre $\pm I_{\max}$ et $\pm D_{\max}$. K_p, T_i et T_d représentent respectivement le coefficient proportionnel, le temps d'intégration et le temps de dérivation du correcteur ;

② `CA-CO : pid-frein.` La lecture de la pression est effectuée dans cette TM.

Résultats des tests En fixant les paramètres de la loi de commande comme suit : ($K_p = 5$; $T_i = 0.8$; $T_d = 0.4$; $K_{i\,\max} = 0.1$; $K_{d\,\max} = 0.2$), et les périodes d'activation des TMs à ($T_{pid-frein} = T_{pression} = 10\ ms$), on obtient la réponse indicielle donnée par la figure 3.1-5(c). On observe que le temps de réponse du frein est très long (environ $2\,\sec$). La moitié

de ce temps est dû aux jeux mécaniques entre l'actionneur et la pédale. Pour améliorer le système de freinage, il faut agir directement sur le circuit hydraulique du freinage.

3.2 Spécification du suivi avec une seule TR

Une spécification informelle serait : "Un véhicule équipé d'un capteur de position va suivre un autre véhicule" [APE95, AEP96b]. L'approche ORCCAD traduit l'action du suivi comme :

- des commandes latérale et longitudinale ;
- la surveillance du démarrage et l'arrêt des asservissements ;

3.2.1 Spécification en temps continu de la partie réflexe de l'action

Les lois de commande (latérale et longitudinale) représentent la partie réflexe de l'action. D'une part la loi de commande longitudinale prend l'accélération A_d comme variable de contrôle des moteurs de locomotion et de freinage. D'autre part la loi de commande latérale prend comme variable de contrôle la position du volant α_c pour le contrôle du moteur de direction. Ces deux variables devront évoluer en fonction de la position du véhicule qui précéde le véhicule à commander. Donc les asservissements des moteurs seront enrichis par les informations du capteur de position présente dans la section 2.3.1 du chapitre 2.

Génération de la consigne longitudinale

La formule donnant la consigne longitudinale A_d en fonction des informations extéroceptives (voir démonstration en Annexe C) :

$$A_d = h^{-1}(dv + C_p(dx - hV_2 - d_{\min})) \tag{3.2-1}$$

où :

- dx est obtenue directement par le capteur de position. Elle représente la distance relative entre les deux véhicules ;
- dv est la vitesse relative estimée par un filtre de Kalman ;
- $d_{\min}$ est la distance de sécurité. Elle représente aussi la distance à l'arrêt et la distance minimale de visibilité de la caméra ;
- V_2 est la vitesse du véhicule commandé. Elle est estimée à partir des mesures odomètriques ;
- h^{-1} est le temps désiré qui sépare les deux véhicules. On le fixe suivant les performances du véhicule.

– le gain C_p est variable en fonction de la vitesse du véhicule V_2 :

  – si l'accélération demandée est suffisante pour donner à la voiture la vitesse demandée (c'est à dire si $V_2 \leq hA_{\max}$), le gain sera : $C_p = h^{-1}$;
  – sinon, il faut le baisser en fonction de la vitesse : $C_p = \frac{A_{\max}}{V_2}$.

Le calcul de la consigne est intégré dans la TM `gt-loc` (figure 3.2-6(a)). Il pourra être récupéré par le contrôleur d'asservissement du moteur de locomotion présenté dans le paragraphe 3.1.1.

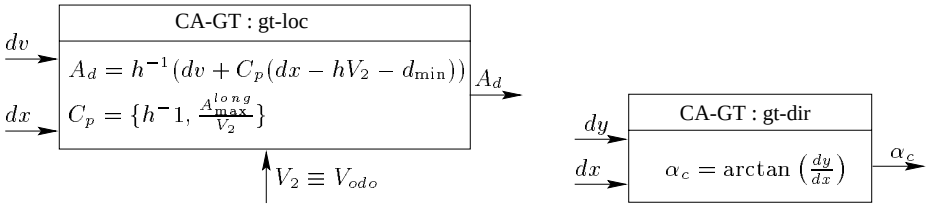
Génération de la consigne latérale

En première version, l'algorithme de la génération de la consigne consiste à pointer les roues avants du véhicule suiveur sur les roues arrière du véhicule suivi. De la même manière que précédemment, le calcul de la consigne α_c est déduit directement à partir des informations données par le capteur de position :

$$\alpha_c = \arctan\left(\frac{dy}{dx}\right) \tag{3.2-2}$$

La consigne est récupérée par le contrôleur du volant présenté dans la section 3.1.2 pour asservir le moteur du volant. L'algorithme de calcul est intégré dans la TM `gt-dir` présentée dans la figure 3.2-6(b).

D'autres modes de suivi latérale plus performants qui mémorisent la trajectoire du véhicule précédent font l'objet de la thèse de [Dav98].



(a) Génération de A_d par GT-LOC. (b) Génération de α_c par GT-DIR.

FIG. 3.2-6 – *Les TMs pour le génération des consignes longitudinale et latérale.*

3.2.2 Spécification du comportement logique

La spécification du comportement logique de la TR consiste à détailler les événements à considérer, la durée de leurs chiens de garde, s'ils existent, et les traitements associés.

– deux événements de type pré-condition (bonne initialisation et ordre de démarrage) ;

– deux événements de type 1 :

  – si $V_2 > hA_{max}^{long}$ changement du gain proportionnel dans le contrôle longitudinal $C_p = \frac{A_{\max}^{long}}{V_2}$;

  – si $\left|\frac{V_2^2 \alpha_c}{l_2}\right| > A_{\max}^{lat}$ alors prendre $\alpha_c = \frac{A_{\max}^{lat} l_2}{V_2^2}$;

– trois événements de type exception avec un traitement associé de type 3 (erreur, données non reçues et changement de mode) ;

– une post-condition (durée de l'expérimentation).

3.2.3 Implémentation

Cette partie constitue l'étape de passage de la spécification de la TR, donc de la partie réflexe et de la partie réactive, à une description qui prend en compte les aspects liés à l'implémentation.

Partie réflexe Le passage consiste à spécifier les TMs de type algorithmique dont les fonctionnalités spécifiques pour les robots mobiles à roues ont été établis. Ces tâches sont *périodiques*. La période dépend de la fonction de cette tâche dans le schéma d'asservissement. La figure 3.2-7 donne une idée de la cadence des TMs suivant la classification faite auparavant. Par exemple, l'estimateur des forces extérieures (TM CA-EST), nécessite une fréquence de rafraîchissement moins grande que celle qui traite l'acquisition des capteurs (TM AQ-PR).

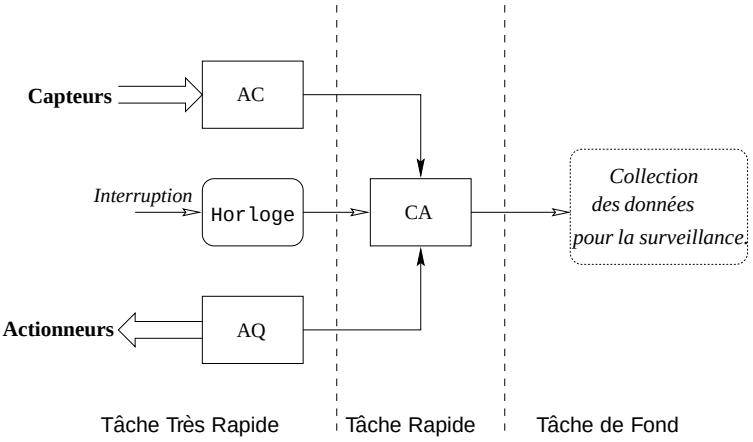


FIG. 3.2-7 – *Les cadences suivant la classification des TMs.*

Cette spécification est complétée par la description de l'ensemble des ports de communication avec le type de synchronisation, les codes d'initialisation et de calcul, le numéro de processeur sur lequel vont être éxécutés ces codes d'initialisation et de calcul.

Partie réactive Elle définit la relation entre les événements d'entrée et un comportement en sortie. Tout le code de cette partie peut être automatiquement produit. La TM représentée par l'objet ATR est *apériodique*, elle est activée à chaque réception d'un signal issu d'une TM de type observateur.

3.2.4 Résultats Expérimentaux

La figure 3.2-8 montre un suivi sur une ligne droite. Le contrôle longitudinal a une vitesse allant jusqu'a $10\ m.s^{-1}$. L'erreur en distance est plus importante dans les phases d'accélération et de décélération mais sans dépasser $0.5\ m$. Le rapport dx/V_2 nous donne la constante de temps $h = 0.4$.

La consigne générée par la TM GT-LOC agit sur le moteur de locomotion pour freiner en récupération. Ceci permet d'avoir une décélération maximale $0.2\ g$. Cette valeur peut être augmentée en utilisant les freins de route.

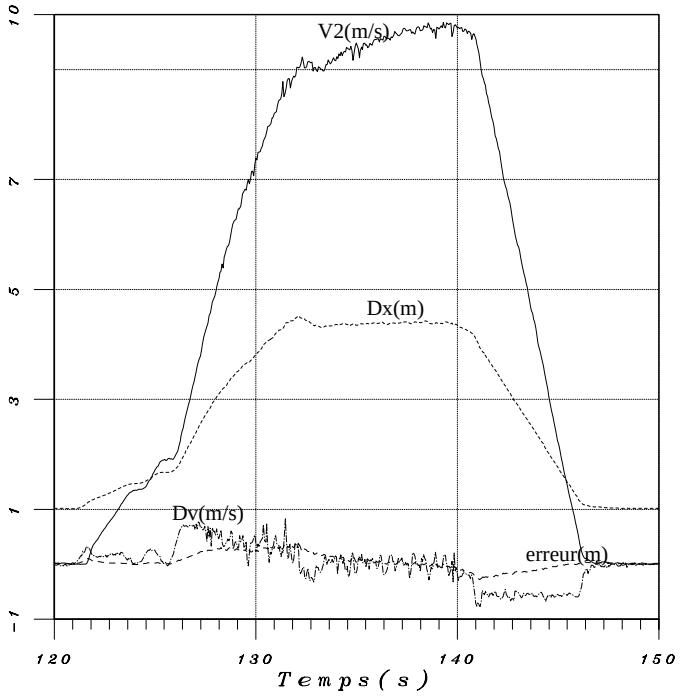


FIG. 3.2-8 – *La TR de suivi sur une ligne droite.*

3.3 Spécification en PRs

La mission du suivi peut être améliorée en :

– utilisant les freins de route. En effet, la consigne donnée par la TM GT-LOC peut

TAB. 3.3-1 – *Spécification des TRs.*

RobotTask	Post-cond	T1-excep	T2-excep	T3-excep
SENSLOC		TagetLost300ms TargetLost1s SatuAcclong	TargetLost StopEmergency	Auto2man MotLocFaill
SENSDIR		SatuAcclat	TagetLost StopEmergency	Auto2man MotDirFaill
BRAKEMAX	OkMax			Auto2man MotBrakeFaill
BRAKEMIN	OkMin			Auto2man MotBrakeFaill
BRAKE				Auto2man MotBrakeFaill

dépasser la limite du freinage moteur $A_{\max}^{long}$. Il est nécessaire de freiner simultanément avec les moteurs de locomotion et de frein de route dans de telles situations ;

– traitant plus finement les pertes de données. Au lieu de s'arrêter dès que la caméra ne voit plus la cible, on va essayer de faire un "suivi en aveugle" pendant quelques ms avant d'arrêter le véhicule. Cela permettra de passer des obstacles simples tels que les dos d'âne, et de ne pas s'arrêter trop souvent.

Par rapport à la spécification avec une TR, le véhicule sera plus autonome en se prenant en charge dans des situations plus compliquées. La programmation à ce niveau nécessite l'utilisation de plusieurs TRs : c'est la programmation au niveau tâche [KAJE95, EJAK95].

3.3.1 Les TRs nécessaires pour le "train" de véhicules

On pourra avoir les TRs données dans la table 3.3-1 :

– SENSLOC est la TR qui agit sur le moteur de locomotion pour effectuer le contrôle longitudinal. Dans son déroulement normal, les données nécessaires au calcul de la consigne A_d proviennent directement du capteur de position. Les événements qui peuvent venir "perturber" le déroulement normal de cette action sont les suivants :

– l'événement `SatuAccLong?` permet de faire le changement du gain de position C_p de la commande en fonction de l'accélération maximale longitudinale ;

– `TargetLost300ms?` est un événement qui arrive à chaque fois qu'il y a des pertes de données pendant une durée de $300\ ms$. Les données capteur dx et dv nécessaires au calcul de la consigne A_d sont estimées. L'estimation est faite sous l'hypothèse que la vitesse du véhicule est constante pendant ce très court intervalle de temps ;

– `TargetLost1s?` est un événement qui survient obligatoirement après `Target-Lost300ms?`, puisqu'il est valide lorsque la cible est cachée pendant $1\ s$. Dans ce cas, un freinage moteur maximal est demandé en substituant dx et dv, par respectivement $d_{\min}$ et V_2 ;

Le traitement des exceptions `SatuAccLong?`, `TargetLost300ms?` et `Target-Lost1s?` consiste en un changement de paramètres dans la loi de commande (équation (3.2-1)), c'est à dire le code de la TM GT-LOC.

–SENSDIR est la TR qui effectue le contrôle latéral. Le calcul de la consigne α_c nécessite aussi les données dx et dy. Lorsque la caméra est cachée, le calcul de la consigne est fait sur la base des dernières valeurs reçues. Le seul événement qui sera pris en compte, est l'événement `SatuAcclat?` qui permet de limiter la consigne du volant en fonction de l'accélération latérale maximale.

– L'action de freinage peut être décomposée en trois TRs. Suivant la situation, le frein peut-être :

– enlevé : BRAKEMIN ;

– mis au maximum : BRAKEMAX ;

– une fonction de la décélération demandée : BRAKE(λ) avec $\lambda = -A_d - 2$. Cette TR est lancée à partir de l'observation de la consigne A_d ;

– supprimé : BRAKEMIN.

Spécification en PRs

Méthodologie On part toujours d'une action principale définissant le but à atteindre. Cette action traduit le déroulement normal de l'application. Puis, on énumére les évènements qui perturbent son bon déroulement et on leurs associe des actions de récupération. Chaque récupération a un sous-but mais sa finalité reste de contribuer à atteindre le but principal. Donc, ces actions s'arrêtent si le sous-but est atteint où bien si les conditions de la reprise de l'action principale sont réunies. Que l'action soit principale ou de récupération, elle peut être une TR ou une PR. Bien sûr, avant que la PR ne soit appelée, il faut qu'elle existe déjà. Par conséquent la première PR ne peut être formée que par des TRs.

Description de la mission de suivi La mission commence après réception de l'ordre de l'opérateur `Start?`, seulement en mode automatique `AutoMode?` et après une bonne initialisation de la voiture `OkInit?` : ce sont les pré-conditions de la PR MAIN.

```
Main % Le debut du programme
Pre-condition  OkInit[30ms],AutoMode[30ms],Start[5mn];
Signal TargetFound[3mn];
T3-exceptions Auto2man, MotFaill;
Post-cond  Stop [60mn]
BeginBody
        Loop
           Seq
              start (BRAKEMAX);
               wait TargetFound;start (BRAKEMIN);
               start (FOLLOWME);
           EndSeq
        EndLoop
EndBody
              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr FOLLOWME; % La PrR FollowMe
Signal MoreBrake;
T2-exception TargetLost, StopEmergency;
BeginBody
        Parallel
           start (SENSLOC)
           start (SENSDIR)
           Loop
              if  MoreBrake then start (BRAKE)
           EndLoop
        EndParrallel
EndBody
BeginRecovery
        if  TargetLost then start (BRAKEMAX)
        if  StopEmergency then start (BRAKEMAX)
EndRecovery
```

Algorithmme 7: Spécification du suivi pour une traduction.

En attendant que la caméra trouve la voiture à suivre, le frein de route est mis légèrement (TRBRAKEGUARDE). Dès que la cible est trouvée, `TargetFound?`, le frein est enlevé (TRBRAKEMIN) et la PR FOLLOWME est lancée. Cette dernière lance les TRs SENSLOC et SENSDIR en parallèle. Pendant son exécution nominale les données caméra ainsi que la vitesse de la voiture sont surveillées pour changer les paramètres des TRs SENSLOC et SENSDIR. Plus explicitement, si la cible est invisible pendant $300ms$ (`TargetLost300ms?`) les données nécessaires pour générer la trajectoire sont estimées et si la cible n'est toujours pas retrouvée après $1s$ (`TargetLost1s?`), la consigne en accélération est choisie pour faire un freinage moteur maximal ($d_x = d_min$ et $dv = V_2$). Par contre, si la cible est retrouvée ce sont les données de la caméra qui seront prises. Un

événement interne lié à la décélération demandée (`MoreBrake?`) lance la TR de freinage hydraulique BRAKE(λ). Les signaux `TargetLost?` et `StopEmergency?` lancent la TR BRAKEMAX : ce sont des signaux d'exceptions de type 2.

3.3.2 Traduction et vérification des PRs

La spécification illustrée dans l'algorithme 7 est traduite manuellement vers le langage synchrone ESTEREL. Son compilateur vérifie les erreurs de causalité et produit un automate à états finis en format `OC`. Il est alors possible d'utiliser des outils de preuve sur ce type de graphe.

L'automate que l'on a obtenu décrit l'évolution de l'application en terme d'actions robotiques. On va vérifier si le comportement de l'application est conforme à notre spécification. Pour cela, le logiciel AUTO est utilisé pour faire la vérification formelle sur l'ensemble des comportements possibles de cette application.

Propriété de sûreté La propriété de sûreté permet, *en cas de problème*, de vérifier que le système est mis dans une *situation sans risque*. La détection d'un problème est traduite par une exception de type 3 et la situation sans risque la plus évidente est l'arrêt. Donc il suffit de vérifier que les signaux d'entrées qui traduisent une exception de type 3 sont toujours accompagnés par le signal de sortie qui permet d'arrêter tout.

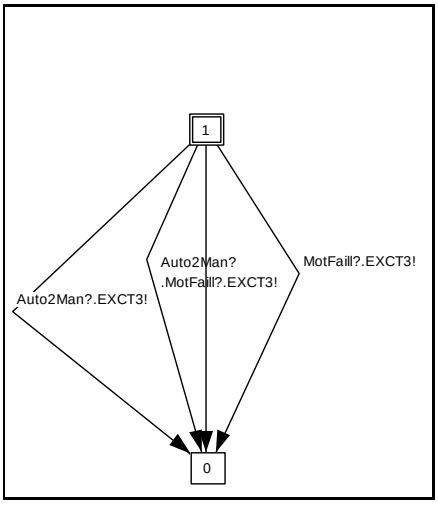
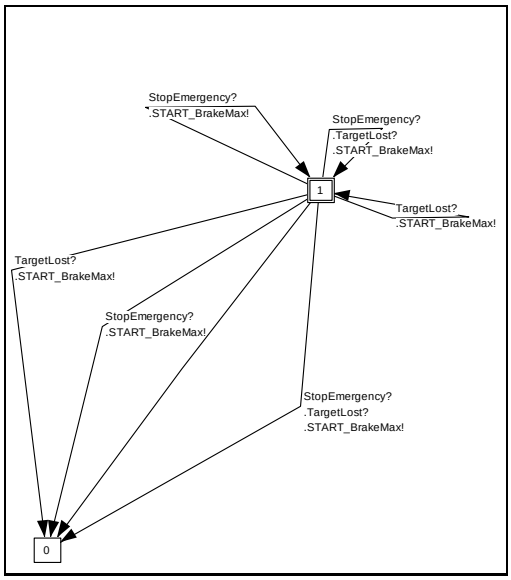
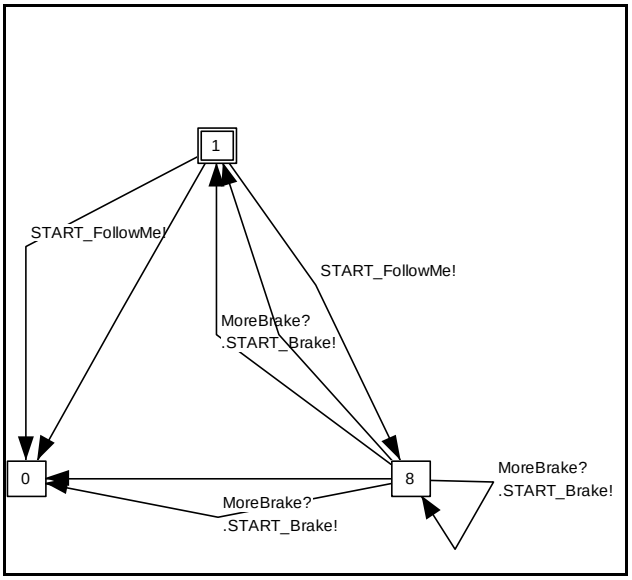


FIG. 3.3-9 – *Propriète de sûreté.*

La figure 3.3-9 représente le résultat de l'observation de cet automate par rapport aux signaux d'entrées de type 3 (`MotFaill?`, `Auto2man?`) et du signal de sortie qui doit être émis (`EXCT3!`). La taille de l'automate est réduite ce qui nous permet de vérifier par simple visualisation que toutes les transitions sont étiquetées d'un signal d'entrée d'exception 3 et d'un signal de sortie qui arrête les asservissements. Ce qui permet de dire que l'action d'arrêt de la voiture est émise à chaque fois qu'un signal indiquant une défection dans le système est présent : la propriété de sûreté est vérifiée.



(a) Les situations où le frein est mis au maximum.



(b) La TR BRAKE commence avec l'événement *Target-Lost*.

FIG. 3.3-10 – *Quelques vérifications sur la PR* FOLLOWME *(i).*

Cohérence de la spécification La spécification en PR et les contraintes imposées par l'application peuvent être aussi vérifiées.

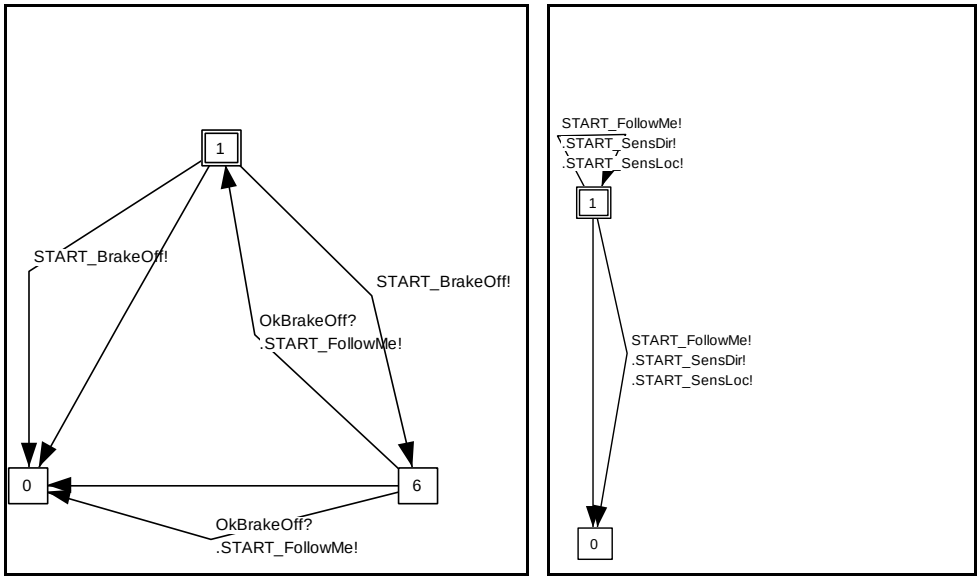
Evénement/Action L'observation de l'automate par rapport aux signaux suivant : `TargetLost?`, `StopEmergency?` et `START_BrakeMax!` (figure 3.3-10(a)), permet de dire que le frein de route est mis au maximum si la cible est perdue ou si l'arrêt d'urgence est demandé. On remarque la prise en compte du cas où ces deux signaux arrivent au même moment..

De la même manière, on peut vérifier par simple visualisation de la figure 3.3-10(b) que la TR BRAKE est lancée par le signal `START_Brake!` si la décélération désirée ne peut pas être satisfaite par la récupération d'énergie du moteur de locomotion (événement `More- Brake?`).

La figure 3.3-11(a) met en évidence la propriété suivante : le suivi ne peut commencer `START_FollowMe!` que si le frein de route est complètement enlevé `OkBrakeOff?`.

Action/Action La figure 3.3-11(a) illustre l'opération de séquencement des TRs BRAKEOFF et FOLLOWME. Ceci vérifie la cohérence avec la spécification, c'est à dire : le frein de route est toujours enlevé `START_BarkeOff!` avant que le commencement du suivi `START_FollowMe!`.

La figure 3.3-11(b) montre que les signaux `START_SensDir!` et `START_SensLoc!`, qui contrôlent le démarrage des TRs SENSDIR et SENSLOC, sont émis simultanément suite



(a) Le suivi ne peut commencer que si le frein est enlevé.

(b) Mise en parallèle des TR SENSLOC et SENSDIR.

FIG. 3.3-11 – *Quelques vérifications sur la PR* FOLLOWME*(ii).*

à la demande faite par le signal `START_FollowMe!` pour démarrer la PR FOLLOWME. Notons que tous ces signaux sont activés au même instant grâce à la diffusion synchrone.

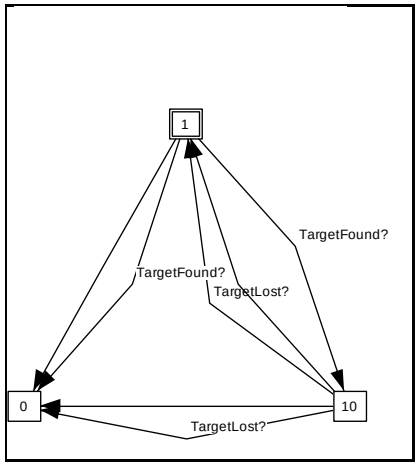
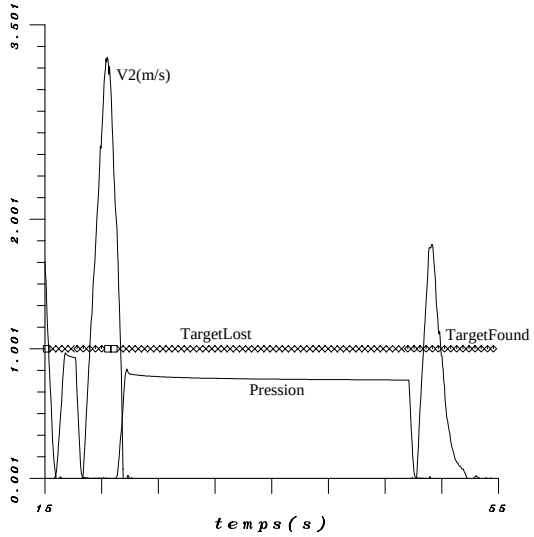


FIG. 3.3-12 – *Bouclage des événemets.*

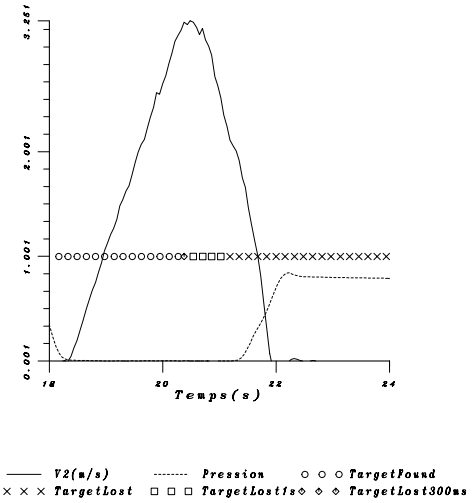
Evénement/événement Dès que la cible est perdue, la PR MAIN attend que la cible soit retrouvée et inversement, figure 3.3-12. Ceci met en évidence la boucle spécifié par l'opérateur `Loop` dans cette PR.

3.3.3 Expérimentations

Plusieurs expérimentations ont été réalisées pour essayer de mettre en évidence diverses situations.



(a) Bouclage des événements *TargetLost* et *TargetFound*.



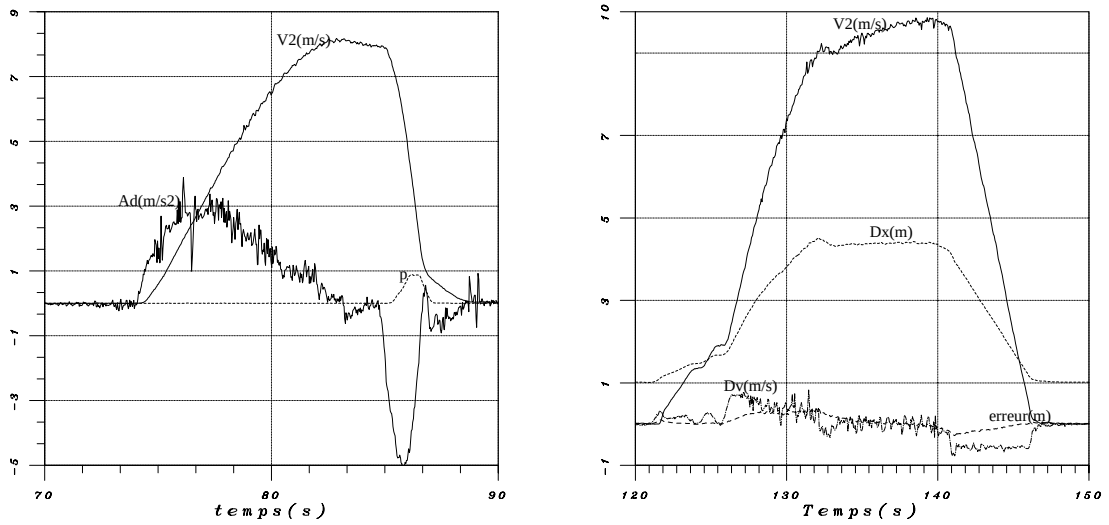
(b) Séquencement des événements *TargetLost300ms*, *TargetLost1s* et *TargetLost1s*.

FIG. 3.3-13 – *Expériences à basse vitesse.*

La première expérience est faite à petite vitesse en cachant la cible du champ de la caméra pendant quelques secondes pour tester la réaction du véhicule. La courbe 3.3-13(a) montre deux accrochages suite à la perte de la cible. Ceci traduit le bouclage des événements `TargetLost?` et `TargetFound?`.

La deuxième expérience est aussi effectuée à basse vitesse. Elle met en évidence le changement de paramètre dans la loi de commande longitudinale pour freiner avec le moteur, si la cible est perdue pendant $300\ ms$. Ensuite, un freinage avec les freins de route (augmentation de la pression p) est lancé lorsque la cible n'est toujours pas retrouvée après $1\ s$ (figure 3.3-13(b)).

La troisième expérience est effectuée à une vitesse proche des $30\ km.h^{-1}$, l'événement `MoreBrake?` apparaît lorsque la décélération demandée est inférieure à $-2\ m.s^{-1}$. Ceci est traduit par une demande de freinage de route en fonction de cette décélération (figure 3.3-14(a)).



(a) Freinage suite à l'événement *MoreBrake*.

(b) Déroulement normnale.

FIG. 3.3-14 – *Expériences à vitesse élevée.*

Dans la quatrième expérience le suivi est réalisé à une vitesse de $36\ Km.h^{-1}$ sans changement de mode (figure 3.3-14(b)).

3.4 Conclusion

Dans ce chapitre nous avons montré comment un train du véhicules sans lien matériel peut être spécifié, vérifié et implémenté. Par conséquent, ce système peut être utilisé comme un moyen pour résoudre le problème du retour à vide des véhicules comme cela était indiqué dans la version 2 du projet PRAXITÈLE. Cette version permet une évolution entièrement autonome des véhicules. En effet grâce au capteur de localisation les véhicules n'ont besoin ni de communication ni de références au sol. Contrairement aux systèmes existants.

L'originalité de la spécification de cette mission sous ORCCAD réside dans le fait de l'utilisation d'une approche cohérente et formelle dans la robotique. Et ceci grâce au langages synchrones. Dans cette mission du suivi, nous n'avons traité que la vérification logique, cependant la vérification des propriétés quantitatives temporelles temps réel est également possible [Jou95].

Nous avons montré à travers un exemple réel et relativement complexe, une approche de programmation modulaire dans la conception des lois de commandes à l'aide des TRs. A partir des "briques" de bases (TMs), nous avons pu réaliser toutes nos TRs. Ce style de programmation nous a permis de passer du mode de suivi sur cible active, vers le suivi sur catadioptre [PDDM94] ou encore pour une conduite avec un "joystick" [BP97] en ne

changeant qu’une seule TMs.

Chapitre 4

Manœuvre de Parking Parallèle

UN système autonome, qui doit effectuer des actions, est amené à se déplacer pour exécuter ses tâches ; il doit donc décider d'un mouvement dans un espace qu'il lui faut *connaître* mais aussi *comprendre*. C'est l'élaboration d'un planificateur de trajectoire qui permet de proposer un plan d'action à exécuter. Dans ce chapitre, nous allons montrer à travers un exemple de manœuvre de parking automatique une manière de spécifier une mission permettant le passage à la version 2 du programme PRAXITÈLE. Une spécification informelle de la mission peut être : "Un véhicule disposé à proximité d'une place libre, va se garer parallèlement". Ceci suppose que le véhicule est disposé devant une place, et que cette dernière est suffisamment spacieuse pour effectuer les manœuvres. Le véhicule doit se prendre en charge à partir de cet instant, en supervisant l'exécution. Par rapport à l'application précédente où le véhicule ne fait que suivre les "mouvements" d'un autre, celle ci nécessite une *stratégie* dans la génération de la trajectoire. La stratégie employée est la suivante :

❶ déplacement en marche avant parallèlement aux obstacles (qui se trouvent sur le coté) sur une longueur prédéterminée. Cette phase permet de construire l'environnement à l'aide des capteurs à ultrasons ;

❷ analyse des mesures pour déterminer si la place est de taille suffisante ;

❸ génération de la trajectoire de parking ;

Une fois la génération de trajectoire faite, nous utiliserons les asservissements des moteurs de locomotion et direction, déjà établis dans le chapitre 3. Le plan de ce chapitre est le suivant :

☞ la section 4.1 présente la technique de localisation, le générateur de trajectoire et les commandes longitudinale et latérale ;

☞ la spécification et la vérification seront détaillées dans la section 4.2 ;

☞ l'enchaînement des deux missions (train de véhicule et parking parallèle) est réalisé dans la section 4.3.

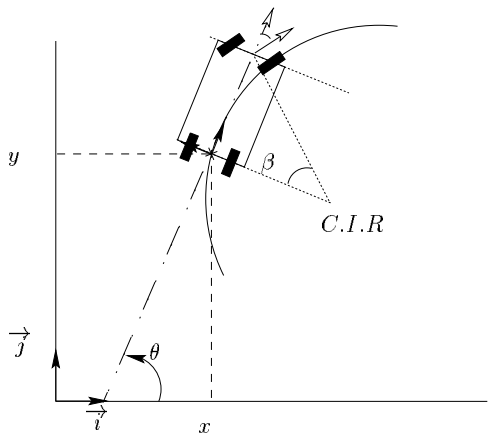


FIG. 4.1-1 – *Mouvement de la voiture par rapport à la trajectoire.*

4.1 Localisation, génération et commande

4.1.1 La localisation de la place de parking

Le véhicule localise la place en parcourant une ligne droite parallèlement à la place. Lors de son déplacement, et tous les quelques centimètres (2 à 10) une mesure de distance aux obstacles à l'aide du capteur à ultrasons est prise. On obtient ainsi un profil de l'environnement qui se trouve sur le coté du véhicule. L'algorithme de recherche de la place se fait en parcourant l'ensemble des données stockées lors de l'acquisition. Il consiste à trouver la distance minimale $D_{\min}$ (figure 4.1-2(a)) plus une marge M correspondant à un créneau. Si la distance à un point est inférieure à $D_{\min} + M$, ce point n'appartient plus au stationnement. Bien sûr, tout ceci suppose que la différence de cap entre le véhicule automatique et les véhicules stationnés n'est pas importante.

4.1.2 Le générateur de trajectoire

La structure mécanique et le rayon de braquage imposent à un robot mobile de type voiture un mouvement tangent à sa direction en un point donné (figure 4.1-1). Dans nos conditions, il n'est pas gênant que le véhicule s'arrête. Les contraintes de continuité de courbe étant levées, notre choix s'est tourné vers des trajectoires composées d'arcs de cercles et de segments de droite.

Reeds et Shepp [RS90] ont montré qu'un nombre fini de familles de chemins de base permettent d'atteindre n'importe quelle configuration. En plus ce chemin est le plus court. Toutefois le générateur de trajectoires présenté ici, est destiné à tester une méthode de programmation et s'intéresse plus particulièrement à certaines manœuvres de parking. Il n'implémente qu'une partie de ces familles pour permettre de se faire une bonne idée des résultats réalisables. Sa réalisation a fait l'objet d'un stage de fin d'étude [Ber95].

Les chemins de base

Le chemin $S(l)$ est un segment de droite de longueur l.

Le chemin $C(R)$ est une portion d'un cercle de rayon R et ayant pour limites la configuration de départ q_0 et la configuration d'arrivée q_1. Une convention sur le signe de R est prise pour différencier le sens de parcourt du chemin :

- $R > 0$ pour un virage gauche. Dans ce cas, le chemin est noté : L ;
- $R < 0$ pour un virage droit. Dans ce cas, le chemin est noté : R ;

Le chemin $C(R_1)S(l)C(R_2)$ est formé de deux cercles, de rayons R_1 et R_2, et d'un segment, de longueur l. Les cercles et le segment sont parcourus dans le même sens.

Le chemin $C(R_1)|S(l)|C(R_2)$ est aussi formé de deux cercles, de rayons R_1 et R_2, et d'un segment, de longueur l. Le segment est parcouru dans le sens opposé aux deux cercles.

Stratégie de parking

Lorsque la place est correctement identifiée, il est nécessaire de construire les contours et les zones utilisées par le générateur. La figure 4.1-2(a) montre les deux zones utilisées, l'une matérialisant la chaussée disponible (zone 1) et l'autre la place de stationnement (zone 2).

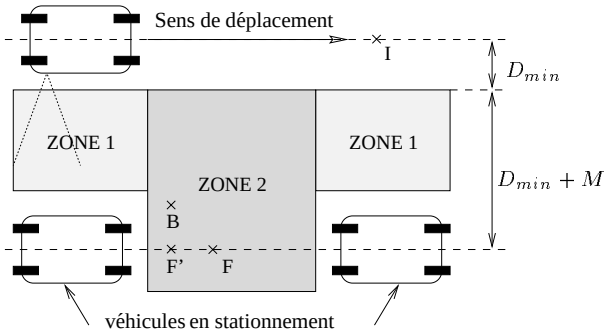
A partir de la position initiale I du véhicule, et connaissant la posture finale F souhaitée, l'algorithme consiste à :

- Premièrement, trouver une position intermédiaire B de **même orientation** que la posture finale F. Le point B est situé dans une "grille" (figure 4.1-2(b)), et doit être accessible depuis I en un seul chemin CSC ;
- Deuxièmement, chercher à rejoindre le point F' à partir du point B en se contraignant à la zone 2. Cette procédure de *translation* est détaillée dans le paragraphe suivant.

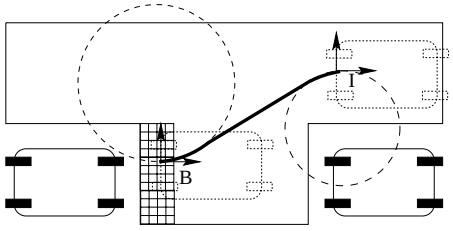
Translation Soit (x_B, y_B, θ) la posture du véhicule au point intermédiaire B et $(x_{F'}, y_{F'}, \theta)$ la posture du point F' que l'on cherche à atteindre en un nombre minimum de manœuvre (figure 4.1-2(c)). L'algorithme de génération des chemins de base pour une translation latérale d'une distance E est donné dans (algorithme 8) où :

- D_E représente la droite qui passe par les points F' et F. Cette droite est située à une distance E du point B ;
- D_e est une seconde droite parallèle à D_E et telle que la distance qui la sépare du point B vaut e. Cette droite devrait être accessible par un seul chemin (RL ou LR) ;

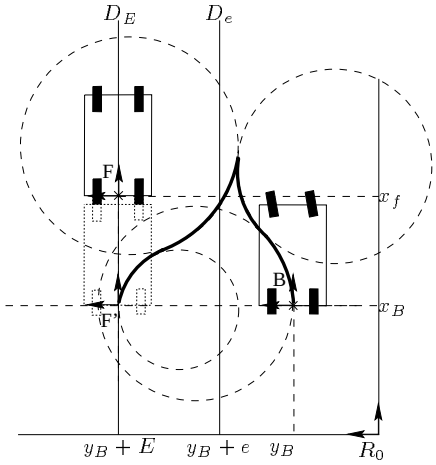
L'algorithme de translation consiste à faire tendre la distance e vers E en utilisant le moins de manœuvres LR et RL. Une fois cette convergence en y atteinte, les manœuvres sont terminées par un segment de droite pour atteindre à partir de F' le point final F.



(a) Localisation de la zone 2 par déplacement en ligne droite.



(b) Placement avec la trajectoire CSC.



(c) Translation avec les chemins LR et RL.

FIG. 4.1-2 – *Stratégie pour le parking.*

4.1.3 Commande du véhicule

La séparation de la commande en deux types de commande, latérale et longitudinale, apparaît naturelle pour le suivi de trajectoire. La commande longitudinale est la même que celle utilisée dans le suivi de véhicule (chapitre 3) avec le véhicule leader à l'arrêt. La commande latérale est inspirée des travaux de C. Samson [Sam93] avec un paramètrage de l'abscisse curviligne différent.

Tant que $E! = 0$ **faire**
  calculer E
  Si il exist une position μ tel que :
    $\mu \in D_E$ **et** μ est accessible par un chemin (RL ou LR)
  alors ajouter ce chemin **et Fin de Tant que**
  Sinon Rechercher e tel que :
    une position $\mu\prime \in D_e$ **et** $\mu\prime$ est accessible par un chemin (RL ou LR) en faisant tendre e vers E.
Fin Tant que
Ajouter un segment pour rejoindre F à partir de $F\prime$

Algorithmme 8: Génération des trajectoires pour une translation.

Commande longitudinale

La commande longitudinale permettra d'atteindre, le plus rapidement possible et sans dépassement, la fin d'un chemin . Supposons qu'un point appartenant à ce chemin à une distance négative ($-s$). Cette distance devient positive ($+s$) s'il y a dépassement. La commande utilisée dans la formation du train du véhicules peut être utilisée comme si le véhicule leader est à l'arrêt. Dans ces conditions, on prend $dx = -s$ (abscisse curviligne) et $dv = -V$, l'équation (3.2-1) s'écrira :

$$A1_d = h^{-1}(-V + C_p(-s - hV)) \tag{4.1-1}$$

avec toujours :

$$C_p = \begin{cases} h^{-1} & \text{si } V \leq hA_{\max} \\ \frac{A_{\max}}{V} & \text{sinon} \end{cases} \tag{4.1-2}$$

Afin d'éviter d'atteindre une vitesse trop élevée sur les longs chemins, un système de saturation de vitesse a été effectué. Une seconde accélération proportionnelle à la différence entre la vitesse maximale $V_{\max}$ et la vitesse du véhicule V_2 est calculée suivant la formule :

$$A2_d = h^{-1}(V_{\max} - V) \tag{4.1-3}$$

Finalement, la commande A_d est choisie comme étant le minimum entre les deux accélérations $A1_d$ et $A2_d$.

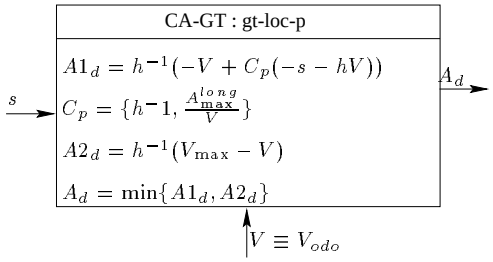


FIG. 4.1-3 – *TM pour la génération de la consigne longitudinale.*

Commande latérale

Modélisation du véhicule La position du véhicule dans le plan est décrite par les coordonnées (x_0, y_0) du point P, milieu de l'essieu arrière et par son orientation θ dans le repère fixe R_0 (figure 4.1-4). Soit V la vitesse du véhicule, β_v l'angle de braquage des roues avant et l_2 la distance entre les essieux avant et arrière. Le véhicule est représenté par le modèle cinématique suivant :

$$\begin{cases} \dot{x_0} = V\cos(\theta) \\ \dot{y_0} = V\sin(\theta) \\ \dot{\theta} = V\frac{\tan(\beta_v)}{l_2} \end{cases} \tag{4.1-4}$$

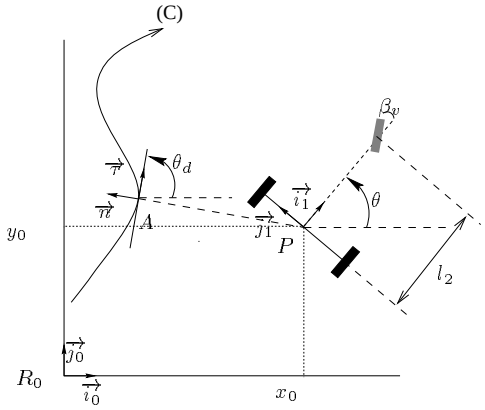


FIG. 4.1-4 – *Suivi de trajectoire.*

Equation dans un repère de Frenet liée à une courbe Pour suivre une trajectoire, on peut soit contrôler l'attitude du véhicule par rapport à un repère fixe soit réguler la distance du véhicule par rapport à la trajectoire désirée. La deuxième solution requiert d'écrire un système d'équations donnant l'évolution du véhicule et les caractéristiques de la trajectoire à suivre. Dans notre cas, les trajectoires à suivre sont simples et différentiables.

Soit (C) la courbe que l'on cherche à suivre et A la projection d'un point P du véhicule sur (C). On associe à ce point un repère de Frenet $(A, \vec{t}, \vec{n})$ lié à la courbe. Ce point est caractérisé par le couple de variables (x, y) où x est la valeur de l'abscisse curviligne au point A et y est le module du vecteur $\overrightarrow{AP}$ tel que : $\overrightarrow{AP} = y\vec{n}$.

L'angle de la tangente à la courbe au point A par rapport à R_0 est θ_d. La variation de cet angle est liée à celle de l'abscisse curviligne x par la relation : $\dot{\theta}_d = c(x)\dot{x}$.

La commande de suivi de trajectoire vise à réguler autour de 0 la distance d'erreur de positionnement latéral y ainsi que l'erreur d'orientation $\tilde{\theta} = \theta - \theta_d$.

Dans le repère de Frenet, les équations (4.1-4) s'écrivent :

$$\begin{cases} \dot{x} = V\frac{\cos\left(\tilde{\theta}\right)}{1-c(x)y} \\ \dot{y} = V\sin\left(\tilde{\theta}\right) \\ \dot{\tilde{\theta}} = V\left(\frac{\tan(\beta_v)}{l_2} - \frac{c(x)cos\left(\tilde{\theta}\right)}{1-c(x)y}\right) \end{cases} \tag{4.1-5}$$

Si on introduit l'abscisse curviligne $s = \int_0^t V dt$ comme paramétre dans les équations, le système 4.1-4 s'écrira :

$$\begin{cases} x' = \frac{dx}{ds} = \frac{\cos\left(\tilde{\theta}\right)}{1-c(x)y} \\ y' = \sin(\tilde{\theta}) \\ \tilde{\theta}' = \frac{\tan(\beta_v)}{l_2} - \frac{c(x)cos\left(\tilde{\theta}\right)}{1-c(x)y} \end{cases} \tag{4.1-6}$$

Commande en $\dot{\beta}_c$ pour réguler y à 0

$$\begin{cases} y' = \sin\left(\tilde{\theta}\right) \\ y'' = \frac{\tan(\beta_v)}{l_2}\cos\left(\tilde{\theta}\right) - \frac{c(x)\cos^2\left(\tilde{\theta}\right)}{1-c(x)y} \\ y''' = \frac{\cos\left(\tilde{\theta}\right)}{l_2\cos^2(\beta_v)}\beta_v' - \frac{\tan^2(\beta_v)}{l_2^2}\sin\left(\tilde{\theta}\right) \\ + \frac{3c(x)}{l_2(1-c(x)y)}\sin\left(\tilde{\theta}\right)\cos\left(\tilde{\theta}\right)\tan\left(\beta_v\right) - \frac{\cos^3\left(\tilde{\theta}\right)}{(1-c(x)y)^3}\frac{dc(x)}{dx} \\ - \frac{3c^2(x)}{(1-c(x)y)^2}\cos^2\left(\tilde{\theta}\right)\sin\left(\tilde{\theta}\right) \end{cases} \tag{4.1-7}$$

Un placement de pôles selon les racines du polynôme $P(\lambda) = (\lambda+\omega_2)(\lambda^2+2\xi\lambda\omega_1+\omega_1^2)$ nous donne les valeurs suivantes :

$$\begin{cases} k_3 = 2\xi\omega_1 + \omega_2 \\ k_1 = \frac{\omega_1^2\omega_2}{k_3} \\ k_2 = \omega_1^2 + 2\xi\omega_1\omega_2 - k_1 \end{cases} \tag{4.1-8}$$

$$y''' + k_3 y'' + (k_1 + k_2)\, y' + k_1 k_3 y = 0 \tag{4.1-9}$$

On en déduit la commande $\dot{\beta}_c$ qui permet de réguler y à 0.

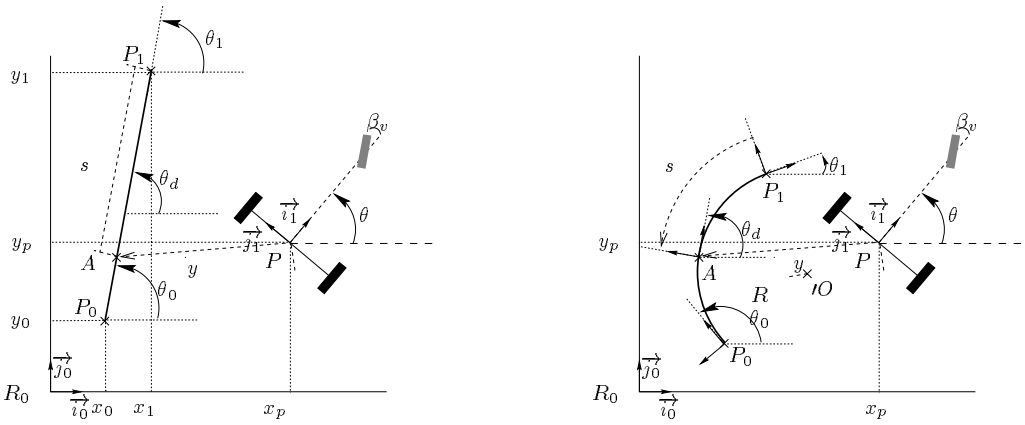
$$\dot{\beta}_c = V\frac{l_2\cos^2\left(\beta_v\right)}{\cos\left(\tilde{\theta}\right)}\left\{\begin{array}{c} y''' + \frac{\tan^2(\beta_v)}{l_2^2}\sin\left(\tilde{\theta}\right) - \frac{3c(x)}{l_2(1-c(x)y)}\sin\left(\tilde{\theta}\right)\cos\left(\tilde{\theta}\right)\tan\left(\beta_v\right) \\ + \frac{\cos^3\left(\tilde{\theta}\right)}{(1-c(x)y)^3}\frac{dc(x)}{dx} + \frac{3c^2(x)}{(1-c(x)y)^2}\cos^2\left(\tilde{\theta}\right)\sin\left(\tilde{\theta}\right) \end{array}\right\} \tag{4.1-10}$$

avec

$$y''' = -k_3\text{Signe}(V)y'' - (k_1 + k_2)y' - k_1 k_3\text{Signe}(V)y \tag{4.1-11}$$

4.1.4 Calcul des consignes longitudinale et latérale $\dot{\beta}_c$ et A_d

Une fois que le générateur de trajectoire a calculé les chemins composant la trajectoire à suivre, il est nécessaire de calculer les variables latérale et longitudinale y et s. Ce calcul dépend de la nature du chemin de la trajectoire. Un module lui est associé. Il a en entrée la liste des chemins de base et en sortie les grandeurs y, s, C et $\tilde{\theta}$ pour calculer les consignes $\dot{\beta}_c$ et A_d. Avant de détailler ces calculs nous introduisons le codage des chemins de base.



(a) Cas d'un segment.

(b) Cas d'un arc.

FIG. 4.1-5 – *Codage des trajectoires.*

Codage des chemins de base Une trajectoire est composée d'un ensemble de chemins de base, et chacun d'eux est entièrement définie par (figure 4.1-5) :

- le type du chemin : segment (seg) représenté dans la figure 4.1-5(a)) ou arc de cercle (arc) représenté dans la figure 4.1-5(b) ;
- le sens de parcourt du chemin : $\varepsilon = \pm 1$;
- la configuration de départ du véhicule : $P_0(x_0, y_0, \theta_0)$;
- la configuration d'arrivée : $P_1(x_1, y_1, \theta_1)$;
- la position du centre et le rayon du cercle portant l'arc : $C(x_{o\prime}, y_{o\prime}, R)$. Avec $R > 0$ si le virage est à gauche, et $R < 0$ s'il est à droite.

Certaines de ces informations sont redondantes mais ceci permet de ne pas les recalculer. La position du véhicule est donnée par $P(x_P, y_P, \theta)$

Distance latérale y C'est la distance entre le centre de l'essieu arrière du véhicule P et le point A (figure 4.1-5) d'un chemin de base.

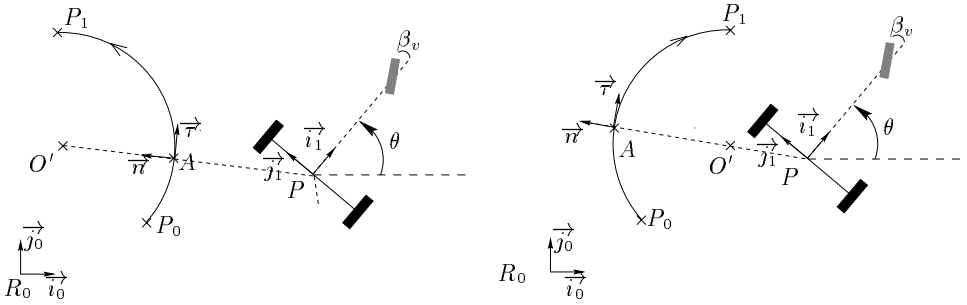
Cas d'un arc de cercle La figure 4.1-6 montre les quatre cas de configurations pour le calcul de cette distance.

Le calcul dépend du sens du virage ($R > 0$ ou $R < 0$) et du sens de déplacement du véhicule sur cette trajectoire (ε). Si le virage est à gauche alors :

$$y = \varepsilon(R - \sqrt{(x_P - x_c)^2 + (y_P - y_c)^2})$$

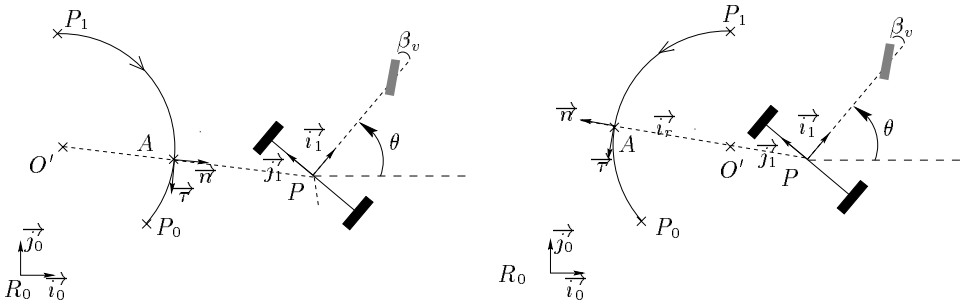
Par contre si le virage est à droite :

$$y = \varepsilon(R + \sqrt{(x_P - x_c)^2 + (y_P - y_c)^2})$$



(a) Virage à gauche en avant : $y = R - \sqrt{(x_p - x_{o\prime})^2 + (y_p - y_{o\prime})^2}$.

(b) Virage à droite en avant : $y = R + \sqrt{(x_p - x_{o\prime})^2 + (y_p - y_{o\prime})^2}$.



(c) Virage à gauche en arrière : $y = -R + \sqrt{(x_p - x_{o\prime})^2 + (y_p - y_{o\prime})^2}$.

(d) Virage à droite en arrière : $y = -R - \sqrt{(x_p - x_{o\prime})^2 + (y_p - y_{o\prime})^2}$.

FIG. 4.1-6 – *Différentes configurations pour le calcul de y pour un chemin de type arc.*

Cas d'un segment Soit A un point appartenant au segment $[P_0, P_1]$ (figure 4.1-5(a)). La distance de ce point au milieu de l'essieu arrière du véhicule est :

$$y = \varepsilon\sqrt{(x_P - x_A)^2 + (y_P - y_A)^2}$$

Nous pouvons exprimer les coordonnées du point A en fonction d'un paramètre K comme suit :

$$\begin{cases} x_A = x_0 + K(x_1 - x_0) \\ y_A = y_0 + K(y_1 - y_0) \end{cases}$$

avec :

$$K = \frac{(x_1 - x_0)(x_P - x_0) + (y_1 - y_0)(y_P - y_0)}{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$

Orientation désirée θ_d On désire avoir le véhicule tangent avec la trajectoire à suivre comme c'est indiqué dans la figure 4.1-5.

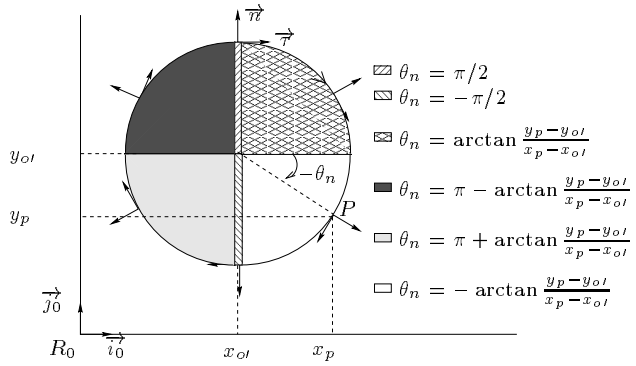
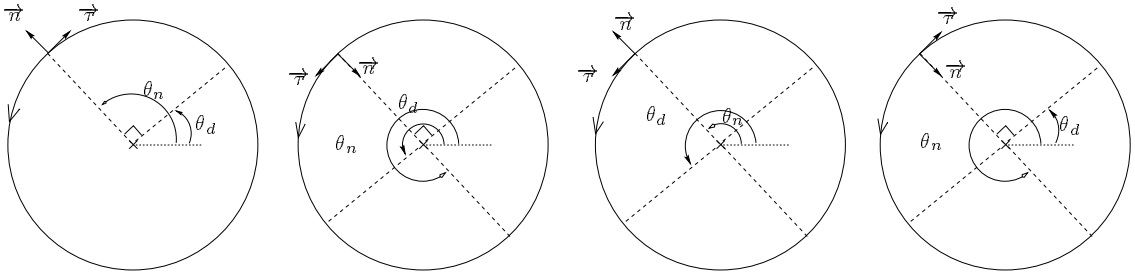


FIG. 4.1-7 – θ_n *lorsque le véhicule avance.*



(a) Virage à gauche en marche arrière : $\theta_d = \theta_n - \pi/2$.

(b) Virage à droite en marche avant : $\theta_d = \theta_n - \pi/2$.

(c) Virage à gauche en marche avant : $\theta_d = \theta_n + \pi/2$.

(d) Virage à droite en marche arrière : $\theta_d = \theta_n + \pi/2$.

FIG. 4.1-8 – *Différentes configurations pour le calcul de* θ_d.

Cas d'un cercle Dans ce cas, le centre instantané de rotation du véhicule devrait être confondu avec le centre du cercle. Pour faciliter la prise en compte de toutes les configurations possibles, nous commencerons par calculer l'angle entre le vecteur unitaire $\vec{n}$ et la position du véhicule représentée par le point P (figure 4.1-7).

Après l'angle θ_d est déduit suivant les différentes configuration illustré dans la figure 4.1-8.

Cas d'un segment Dans ce cas le calcul est immédiat : $\theta_d = \theta_0 = \theta_1$

Distance longitudinale s C'est la distance parcourue par le véhicule le long de la trajectoire.

Cas d'un cercle Connaissant l'orientation désirée θ_d et l'orientation finale θ_1, on peut déduire la distance longitudinale s :

$$s = \varepsilon R(\theta_d - \theta_1)$$

Cas d'un segment Suivant le paramètre K introduit précédemment, la distance longitudinale s est :

$$s = \varepsilon(K-1)\sqrt{(x_P - x_A)^2 + (y_P - y_A)^2}$$

Autres grandeurs nécessaire pour la commande latérale Pour le calcul de la commande latérale suivant l'expression (4.1-10), il manque la courbure C et l'erreur d'orientation $\tilde{\theta}$. La courbure est nulle pour une droite et elle vaut ε / R pour un cercle de rayon R. L'erreur d'orientation est égale à $\theta - \theta_d$ quelque soit la trajectoire.

4.2 Spécification du PARKING

La spécification en PRs peut être décrite avec un langage métier. Ce langage peut être textuel (algorithmes 10) ou graphique (figure 4.2-9). La traduction de cette spécification consiste à produire un programme ESTEREL qui implémente le comportement spécifié. La forme du code produit n'est pas unique, elle va dépendre de la manière dont elle sera interfacée avec la partie décrivant les algorithmes de commande. Nous avons privilégié une approche de spécification qui maximise l'expressivité du code produit. La contre partie est que la taille de l'automate produit est plus importante.

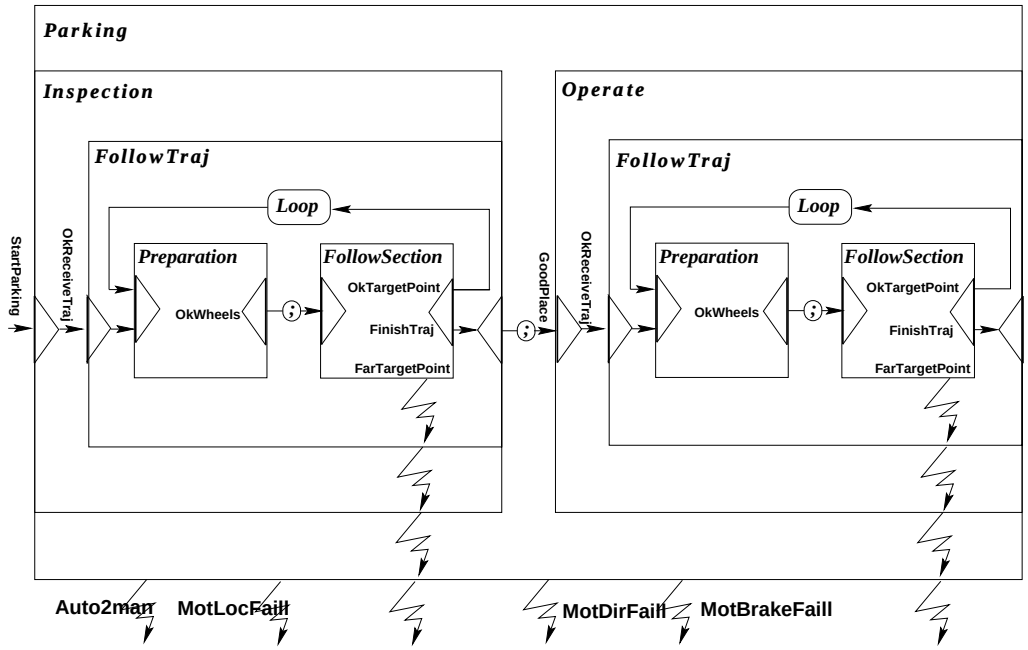


FIG. 4.2-9 – *Illustration sous forme graphique de la mission de parking avec le formalisme des PRs.*

• **une portion contient les informations suivantes :**
type : seg ou arc
Sens : $\varepsilon = \pm 1$
Configuration de départ : $P_0(x_0, y_0, \theta_0)$
Configuration d'arrivée : $P_1(x_1, y_1, \theta_1)$
Position du centre et rayon du cercle portant l'arc :
$C(x_{o\prime}, y_{o\prime}, R)$
$R > 0$ virage à gauche
$R < 0$ virage à droire
• **la posture du véhicule est donnée par :** (x_P, y_P, θ)
Faire
Si type=seg
calculer les coordonnées de $A \in [P_0, P_1]$ tel que :
$dist(P_0, P_1) = 1 \Rightarrow K = \frac{(x_1 - x_0)(x_P - x_0) + (y_1 - y_0)(y_P - y_0)}{(x_1 - x_0)^2 + (y_1 - y_0)^2}$
$x_A = x_0 + K(x_1 - x_0)$
$y_A = y_0 + K(y_1 - y_0)$
$y = \varepsilon\sqrt{(x_P - x_A)^2 + (y_P - y_A)^2}$
$s = \varepsilon(K - 1)\sqrt{(x_P - x_A)^2 + (y_P - y_A)^2}$
$C = 0$
$\theta_d = \theta_0 = \theta_1$
$\tilde{\theta} = \theta - \theta_d$
Fin Si
Si type=arc **faire**
Si $R > 0$ % virage à gauche
$y = \varepsilon(R - \sqrt{(x_P - x_c)^2 + (y_P - y_c)^2})$
Si Non % virage à droite
$y = \varepsilon(R + \sqrt{(x_P - x_c)^2 + (y_P - y_c)^2})$
Fin Si % voir figure 4.1-6
calculer θ_n % voir figure 4.1-7
calculer θ_d % voir figure 4.1-8
$s = \varepsilon R(\theta_d - \theta_1)$
$C = \varepsilon / R$
$\tilde{\theta} = \theta - \theta_d$
Fin Si
Fin faire

Algorithmme 9: Récapitulation des calculs nécessaires pour les commandes longitudinale (s) et latérale (y, C et $\tilde{\theta}$).

D'après la stratégie de parking adoptée, il nous apparaît naturel de décomposer la mission en deux PRs [DAP96] :

– la première PR permet de localiser la place de parking. C'est la PR INSPECTION ;

– et la deuxième pour faire les manœuvres de parking : c'est la PR OPERATE.

Notons qu'au niveau des signaux, les deux PRs ne diffèrent que par l'attente d'une pré-condition `GoodPlace?` dans la deuxième, alors que la première n'a pas de pré-condition. Cette légère différence nous permettra d'utiliser les mêmes modules ESTEREL.

En effet, la PR FOLLOWTRAJ est utilisée dans la PR d'INSPECTION et dans la PR OPERATE. Elle ne peut commencer que si la trajectoire à suivre est bien reçue, cet événement est traduit par le présence du signal `OkReceiveTraj?`. Elle s'arrête si la liste des chemins de base composant cette trajectoire est vide (`FinishTraj?`). Sa terminaison met fin à la PR qui l'englobe (INSPECTION ou OPERATE). Le corps de cette PR est composé de la mise en séquence de deux PRs :

– PREPARATION est une PR qui oriente les roues du véhicule dans la direction tangente par rapport au chemin à suivre. Longitudinalement, elle maintient le véhicule arrêté au point tangent. Une post-condition lui est associée lorsque les roues sont bien placées (`OkWheels?`) ;

– FOLLOWSECTION fait les contrôles latéral et longitudinal. Deux TRs DIR et LOC (table 4.2-1) sont utilisées pour le suivi d'un chemin de base. Un traitement d'exception type 3 est associé lorsque le point du véhicule est trop loin du point de référence `FarTargetPoint?`.

Au lieu d'avoir une spécification textuelle, on peut imaginer une approche graphique (figure 4.2-9). Le sens de l'exécution est de gauche à droite avec une encapsulation des PRs :

– Une PR est représentée par un `rectangle`. Les signaux de pré-conditions sont des signaux entrants. Elles sont représentées par un `triangle` situé à gauche. Par contre, les signaux de post-conditions sont représentés par un `triangle` de sens opposé. Une `ligne brisée` représente une exception de type 3 ;

– Les opérateurs entre les PRs sont représentés dans des `rectangles arrondis` et des `cercles`.

```
Main % Le programme principal
Pre-condition  OkInit[30ms],AutoMode[30ms],StartParking[5mn];
T3-exceptions Auto2man, MotLocFaill, MotDirFaill, MotBrakeFaill;
Post-cond  Stop [60mn]
BeginBody
        Seq
           start (INSPECTION); start (OPERATE);
        EndSeq
EndBody %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr INSPECTION; % La PrR Inspection
BeginBody
        start (FOLLOWTRAJ);
EndBody %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr OPERATE; % La PrR Operate
Pre-condition  GoodPlace[2000ms];
BeginBody
        start (FOLLOWTRAJ);
EndBody %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr FOLLOWTRAJ; % La PrR FollowTraj
Pre-condition  OkReceiveTraj[10ms];
Post-condition  FinishTraj;
BeginBody
        Loop Seq
           start (PREPARATION); start (FOLLOWSECTION);
          EndSeq EndLoop
EndBody %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr PREPARATION; % La PrR FollowTraj
Post-condition  OkWheels;
BeginBody
        Parallel
           start (LOC); start (DIR);
       EndParallel
EndBody %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
RobotPr FOLLOWSECTION; % La PrR FollowSection
T3-exceptions FarTargetPoint;
Post-condition  OkTargetPoint;
BeginBody
        Parallel
           start (LOC); start (DIR);
       EndParallel
EndBody
```

Algorithmme 10: Illustration sous forme textuelle de la mission du parking.

TAB. 4.2-1 – *Spécification des TRs.*

RobotTask	Post-cond	T1-excep	T2-excep	T3-excep
LOC		SatuAccLong SatuVitLong	StopEmergency	Auto2man MotLocFaill
DIR		SatuAccLat		Auto2man MotDirFaill

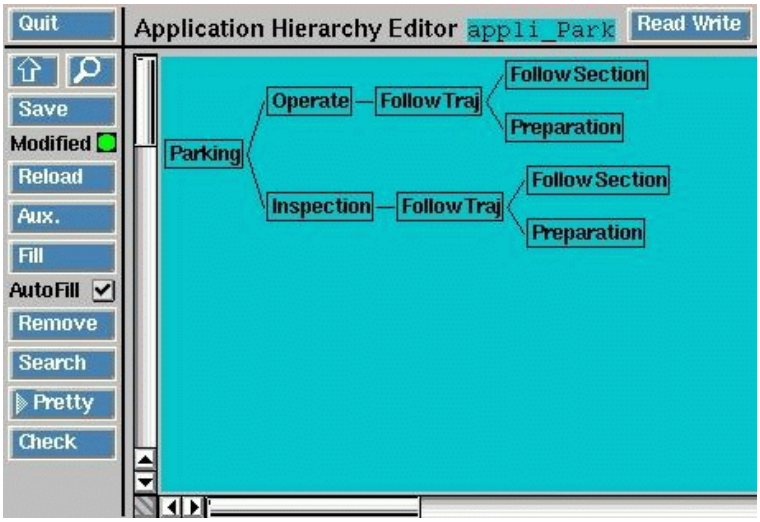


FIG. 4.2-10 – *Les modules* ESTEREL *pour spécifier le parking.*

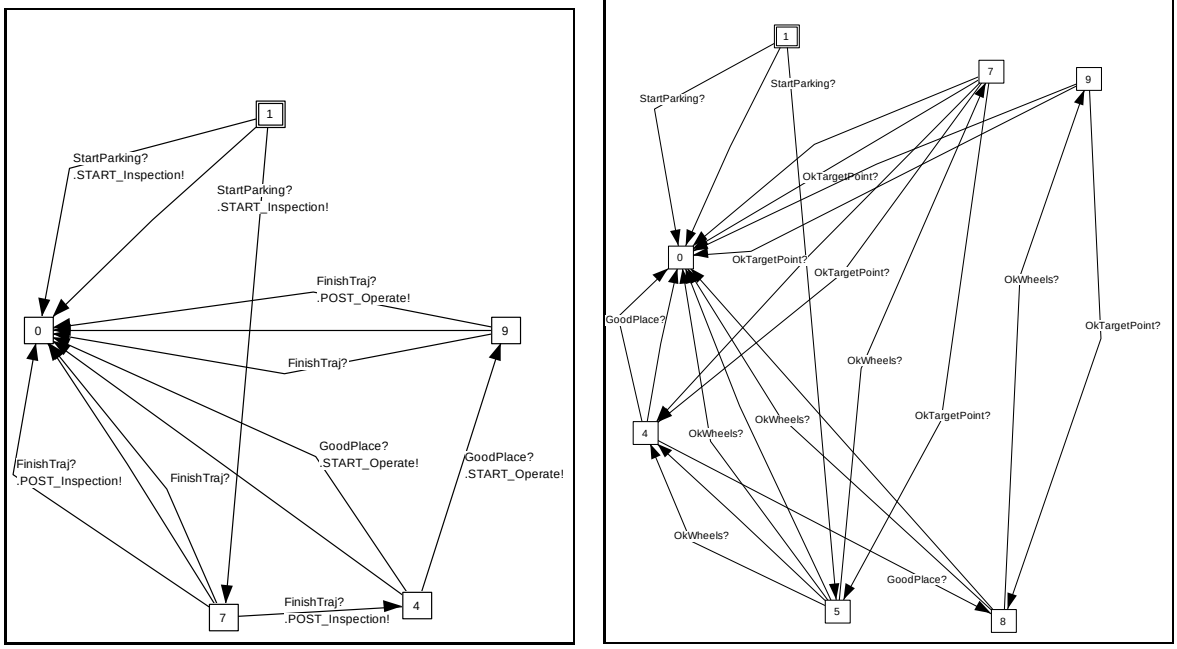
Les deux approches de spécifications (textuelle et graphique) n'ont aucune sémantique et ne permettent pas de produire du code. Elles sont des illustrations permettant de mieux comprendre la spécification avec le formalisme des PRs. La spécification a été traduite manuellement vers le langage ESTEREL. L'architecture du programme traduit est organisée en modules, comme le montre la figure 4.2-10, où chaque module représente une PR. Le problème de la traduction automatique d'une des deux spécifications est traité dans [CMT97].

4.2.1 Cohérence de la spécification

Vérification des PRs INSPECTION **et** OPERATE

Sur la figure 4.2-11(a) on peut vérifier visuellement deux propriétés :

Evénement-Action La PR INSPECTION commence dés que le signal `StartParking?` est reçu. Son démarrage est traduit par l'émission du signal `START_Inspection!`. Et elle se termine avec l'événement `FinishTraj?` par l'émission du signal `POST_Inspection!`. De la même manière, le signal `GoodPlace?` démarre la PR OPERATE et le signal `Finish- Traj?` l'arrête. On voit bien que la caractéristique de programmation qui tient compte de l'ordre d'occurrence des événements, fait que ce dernier signal a une action



(a) Séquence entre PRs INSPECTION et OPERATE.

(b) Mise en évidence de la boucle entre `oKWheels?` et `OkTargetPoint?`.

FIG. 4.2-11 – *Vérification des opérateurs de séquencement et de bouclage.*

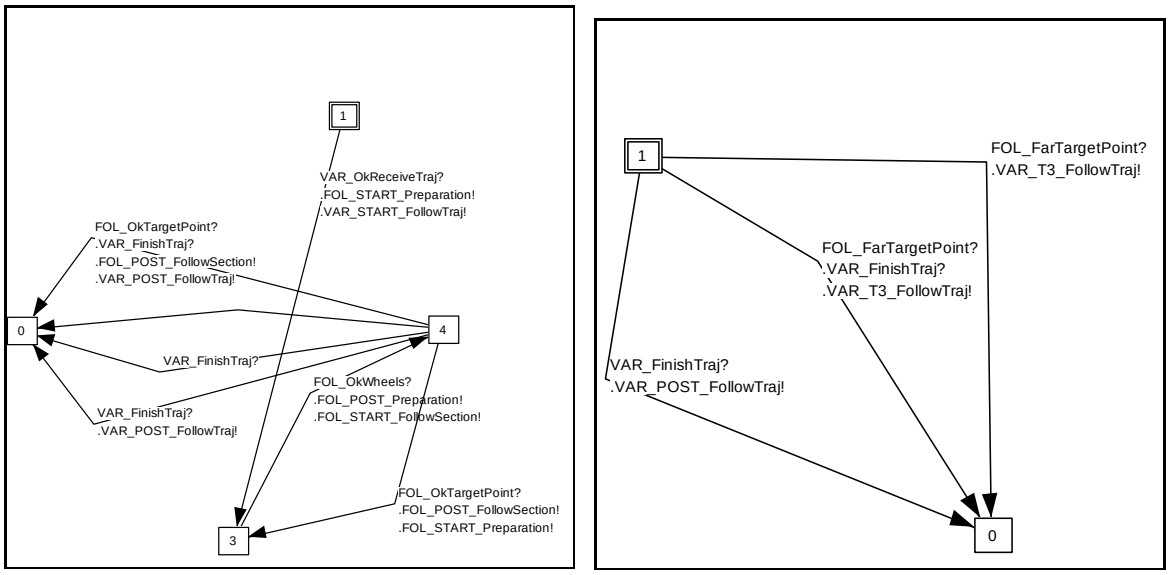
différente dans les deux receptions.

Action-Action L'ordre d'un déroulement normale c'est à dire sans exception de type 3 est représenté par les états : 1, 7, 4, 9, 0. La transition d'un état vers un autre active les actions suivantes : `START_Inspection!`, `POST_Inspection!`, `START_Operate!`, `POST_Operate!`. Ce qui veut dire que les PRs INSPECTION et OPERATE sont lancées séquentiellement.

Bouclage des PRs PREPARATION **et** FOLLOWSECTION

Premièrement, faisons la vérification au niveau de la PR FOLLOWTRAJ. Cette PR est utilisée respectivement dans les PRs INSPECTION et OPERATE. C'est pour cela que ses signaux d'entrées et de sorties sont en paramètres (`VAR_XXX!?`). La vérification de cette PR suivant la classification des propriétés énoncées précédemment, peut être faite par simple visualisation de l'automate réduit de la figure 4.2-12(a).

Evénement-Action La PR FOLLOWTRAJ commence en même temps que la PR PREPARATION par émission simultanée des signaux `FOL_START_Preparation!` et `VAR_START_FollowTraj!`. Elle ne peut commencer que si le signal `VAR_OKReceiveTraj?`



(a) Mise en évidence du bouclage des PRs FOLLOWSECTION et PREPARATION.

(b) Priorité de traitement de l'exception de type 3.

FIG. 4.2-12 – *Quelques vérifications de la PR* FOLLOWTRAJ.

est présent. Cette PR se termine par émission du signal `VAR_POST_FollowTraj!` dès que le signal `VAR_Finish- Traj?` est présent ;

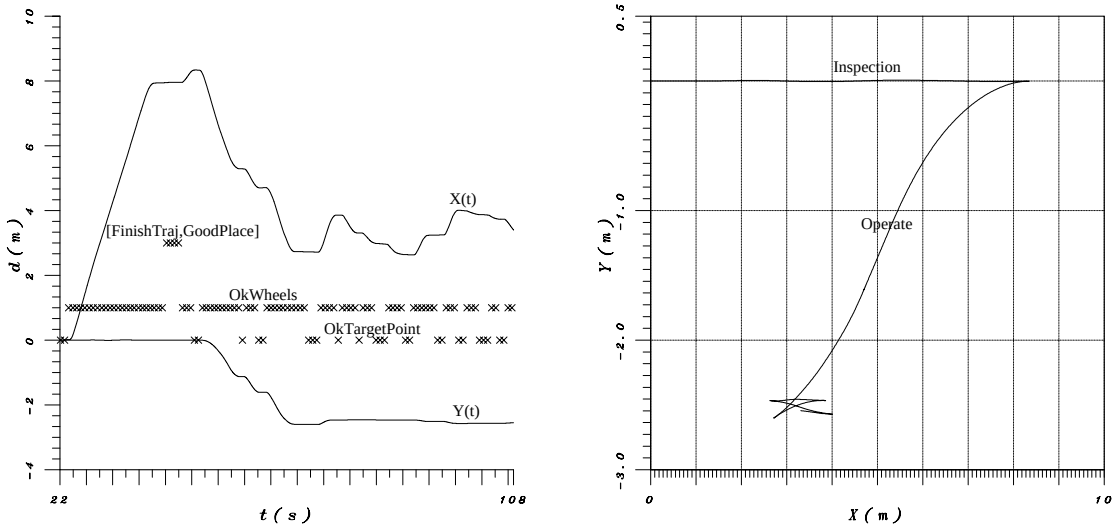
La fin de la PR PREPARATION correspond au commencement de la PR FOLLOWSECTION. Le signal `FOL_OkWheels?` est considéré comme une pré-condition pour la PR FOLLOWSECTION et comme une post-condition pour la PR PREPARATION. De même la fin de la PR FOLLOWSECTION correspond au commencement de la PR PREPARATION. `FOL_OkTargetPoint?` est le signal de post-condition pour la PR FOLLOWSECTION et il est la pré-condition de démarrage de la PR PREPARATION ;

Action-Action Le bouclage des PRs PREPARATION et FOLLOWSECTION est mis en évidence par le couple de signaux (`FOL_POST_Preparation!`, `FOL_START_Follow-Section!`) et (`FOL_POST_FollowSection!`, `FOL_START_Preparation!`) entre les états 3 et 4 ;

Evénement-Evénement entre les états 3 et 4. On voit le bouclage des événements `FOL_-OkWheels?` et `FOL_OkTargetPoint?`.

On peut maintenant se poser la question suivante : est-ce que ces propriétés sont sauvegardées lorsque cette PR est intégrée dans la PR principale de PARKING?

En prenant la PR principale de PARKING, qui contient deux appels de la PR FOLLOWTRAJ, on devrait trouver les deux boucles. En effet, si on visualise l'automate (voir figure 4.2-11(b)) de cette PR relativement aux signaux qui font démarrer la PR FOLLOWTRAJ une première fois `StartParking?` et une second fois `GoodPlace?`, et les signaux qui



(a) Une exécution du parking.

(b) Enchainement des PRs PREPARATION et FOLLOWTRAJ.

FIG. 4.2-13 – *Résultats expérimenataux.*

font la boucle sur les PRs PREPARATION et FOLLOWSECTION (i.e `OkTargetPoint?` et `OkWheels?`), on voit qu'ils y sont entre les états 5 et 7 et entre les états 8 et 9.

Priorité des traitements

Quelle est le traitement le plus prioritaire lorsque deux signaux, l'un traduisant une exception de type 3 et l'autre une post-condition, arrivent au même instant ? Il nous paraît normal de traiter en priorité les signaux d'exception type 3. L'automate réduit de la figure 4.2-12(b) nous montre qu'en présence simultanée des signaux `VAR_FinishTraj?` et `FOL_TargetPoint?`, c'est le signal `VAR_T3_FollowTraj!` qui est émis.

4.2.2 Présentation des résultats expérimentaux

La figure 4.2-13(a) représente une manœuvre de parking. Pendant l'exécution de la PR INSPECTION, $Y(t)$ reste constant pour faire une ligne droite et $X(t)$ augmente constamment en fonction du temps. Après un parcours de $8m$ le véhicule s'arrête durant $5s$. Ce temps correspond à l'instant d'occurrence du signal indiquant la fin du trajet (`FinishTraj?`) et du signal traduisant la possibilité de ce garer (`GoodPlace?`). Le premier signal met fin à la PR INSPECTION et le deuxième lance la PR OPERATE. Cette PR lance séquentiellement les PRs PREPARATTION et FOLLOWSECTION. Le démarrage et l'arrêt de ces deux PRs sont traitées par les signaux `OkWheels?` et `OkTargetPoint?`.

La figure 4.2-13(b) montre bien la stratégie du parking : une ligne droite par la PR INSPECTION (trajet S) puis placement (courbe CSC) et translation (courbes LR et RL)

par la PR OPERATE.

4.3 Enchaînement suivi-parking

Maintenant que nous avons deux grandes PRs PLATOONING et PARKING , nous allons les enchaîner dans une même mission. La décision de la décomposition du *train virtuel* et le commencement de la manœuvre de PARKING du dernier véhicule est prise par un opérateur [AEP96a]. On peut imaginer le scénario suivant :

– Le conducteur donne un signal `Start?` de départ de la mission. Donc de lancement de la PR PLATOONIG ;

– Une fois que le train est arrivé devant une bonne place de parking, le chauffeur donne le signal `StartParking?` pour commencer les manœuvres ;

L'approche de programmation modulaire nous a permis de réaliser cette mission avec moindre effort.

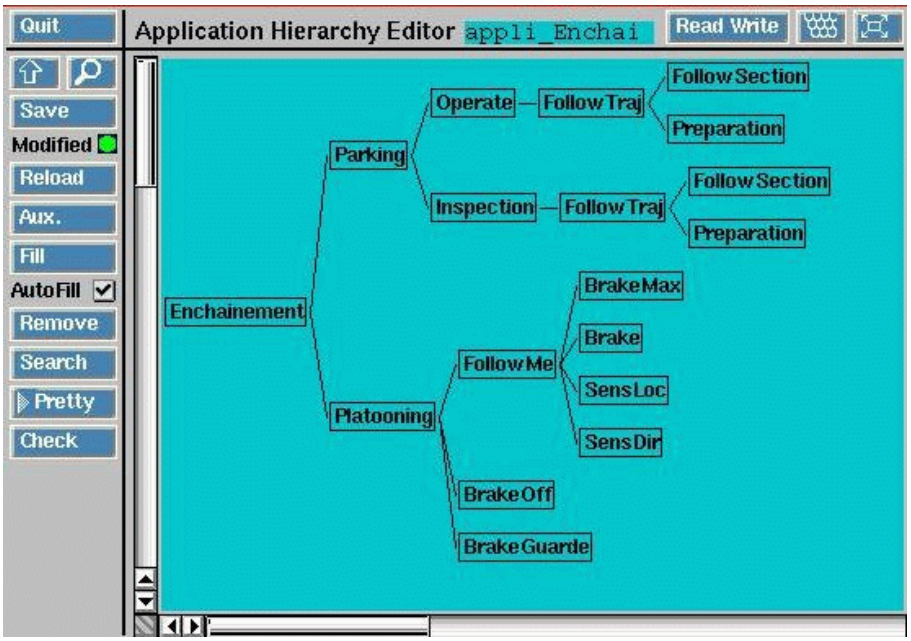


FIG. 4.3-14 – *Les modules* ESTEREL *qui enchainent le suivi et la parking.*

4.4 Conclusion

Les asservissements des moteurs sont les mêmes dans les missions du suivi et de parking, donc les TMs permettant de les faire sont réutilisées. Par contre, les TMs générateurs de la classe (CA-GL) qui calculent les consignes longitudinale A_d et latérale $\dot{\beta}_c$ sont réécrites suivant les expressions (4.1-1) et (4.1-2). Du point de vue performances, la mission

du train de véhicules est plus réactive car les consignes tiennent compte des informations données par le capteur de position (boucle fermée sur ces données). Ce n'est pas le cas dans la mission du parking où les informations des données ultrasonores ne sont pas utilisés pendant l'exécution de la PR OPERATE (un seul plan est exécuté). En contre partie, la mission du parking a une stratégie de planification lui permettant de calculer les chemins de base à suivre. Une amélioration de cette mission a été réalisée dans [VZ96] en ajoutant dans la PR MAIN un événement qui tient compte des obstacles rencontrés pendant les manœuvres.

Finalement, les améliorations de cette mission devront être faites au niveau de la planification. Une application plus réactive a été réalisée sur un deuxième prototype ATALANTE 02 en utilisant les mêmes fonctionnalités de base [PL96]. Dans cette application, une seule TR est utilisée pour effectuer alternativement les asservissements longitudinal et latéral avec bouclage sur les données ultrasonores.

Bilan et Perspectives

LE travail présenté dans cette thèse entre dans le cadre de l'action de développement PRAXITÈLE à l'INRIA Rocquencourt et en collaboration avec le projet BIP de l'INRIA Rhône-Alpes. Cette action s'intéresse à l'automatisation des véhicules électriques afin d'apporter des solutions innovantes à un nouveau mode de transport. Je me suis intéressé à la spécification d'un environnement logiciel permettant la programmation sûre et efficace des fonctions automatisées de ces véhicules. L'environnement ORCCAD développé à l'INRIA Sophia-Antipolis et l'INRIA Rhône-Alpes peut constituer une réponse satisfaisante aux besoins de nos applications. Seulement, les concepts de cet environnement n'ont jamais été validés dans un contexte d'expérimentation réelle. Mes contributions personnelles sont :

- la modélisation du prototype ATALANTE à partir d'une approche générique;
- l'étude de l'influence des erreurs de modèle et de mesures sur l'estimation d'état permettant de cerner le domaine de validité du modèle;
- l'adaptation de l'environnement ORCCAD par la mise en œuvre des fonctionnalités de base de contrôle-commande sur le prototype ATALANTE;
- la spécification, la vérification et l'implémentation de deux missions;

Bilan

Deux objectifs sont à l'origine de ce travail :

- L'étude de la faisabilité des techniques robotiques qui seront utiles dans la réalisation d'un nouveau mode de transport;
- L'utilisation des outils de spécification, vérification et implémentation de l'environnement ORCCAD pour valider l'approche de programmation à travers des exemples très représentatifs.

Les deux points sont en fait très liés. Des applications complexes sont nécessaires pour valider les outils de programmation de recherche; et vice versa : il faudra des outils efficaces pour pouvoir réaliser des applications complexes. Notre but est d'adapter les solutions de la recherche aux contraintes des applications tout en maximisant les **performances.**

Dans le processus de conception d'un environnement de programmation, il faut dans une première phase **valider ses concepts de base** par l'intermédiaire d'outils. Des rectifications peuvent être envisagées si les performances obtenues ne sont pas satisfaisantes. Dans la deuxième phase, il faut maximiser les **services** de l'environnement pour diminuer le temps de développement. Si de nouveaux besoins apparaissent encore pour des applications bien spécifiques, il faut trouver des solutions (concepts et outils) puis revenir à la première phase.

Ainsi le bilan de nos travaux peut être vu sur plusieurs points :

Performances des missions implémentées

Du point de vue des applications visées, les résultats sont satisfaisants puisque nos expérimentations ont fait l'objet de plusieurs démonstrations publiques sans qu'aucun défaut n'apparaisse lié à l'approche de programmation.

Spécification avec l'approche ORCCAD

C'est à partir des exemples tirés des applications spécifiques à PRAXITÈLE que nous avons :

- montré que la classification des signaux caractérisant le comportement logique de la TR, qui à l'origine était faite pour les robots manipulateurs, peut être étendue aux applications de la robotique mobile;

- validé la programmation des missions avec l'entité PRs. La dernière mission, enchaînement de suivi et parking, est constituée d'une trentaine de TMs (écrite à la main). L'automate d'une taille de 19 états, 56 actions et 10810 transitions modélisant la mission en terme d'actions a été produit à partir du compilateur ESTEREL.

- Pour minimiser le code écrit à la main (qui représente une grande source d'erreur) et le temps de développement, une classification des TMs pour la robotique mobile à été faite pour fournir à l'automaticien une librairie d'algorithmes "très bas niveau" permettant d'intégrer les algorithmes de commande.

Perspectives

Les perspectives ouvertes par ce travail sont nombreuses. Nous évoquons ici trois directions :

Spécification-vérification

– Nous avons vu que nos vérifications sont situées au niveau des actions robotiques. Des vérifications, comme l'absence de *blocage temporel* où *deux TMs* n'ont aucun instant d'exécution en commun ou *deux TMs s'exécutent simultanément*, peuvent être faites au niveau de la conception des lois de commande. Ces propriétés pourraient être vérifiées à partir d'une modélisation de la TR avec ARGOS TEMPORISÉ. Des modèles encore plus évolués peuvent être étudiés à partir des systèmes hybrides, comme le font les applications du projet PATH de l'université de Berkeley;

– Les TMs qui ne s'exécutent pas simultanément ne doivent consommer du temps "cpu" que lors de leurs utilisations effectives. Une analyse globale de l'enchaînement des TRs et des TMs qui la constitue permettrait d'optimiser les ressources physiques à utiliser.

– L'utilisation de l'approche synchrone sur les deux niveaux de programmation (TRs et PRs) en utilisant l'approche synchrone permet de faire de la vérification sur les deux niveaux. Les langages synchrones déclaratifs et impératifs permettent de répondre au besoin des deux couches : flot de données pour les algorithmes de commande et flot de contrôle pour leur enchaînement. Des travaux préliminaires ont été entamés pour faire cohabiter les deux styles dans un langage mixte.

Amélioration et nouvelles applications

Amélioration des applications existantes. Parmi les dizaines de démonstrations faites, les quelques échecs ont pour origine des défauts dans un capteur. Les applications pourraient être rendu plus robustes en gérant leurs erreurs;

Réalisation de nouvelles missions avec une plus grande autonomie. Dans la mission de "suivi plus parking" la décision de terminer le suivi et de commencer le parking est prise par un opérateur aprés analyse de la situation (place suffisament spacieuse, ...). Les capacités d'autonomie du robot seront augmentées si un niveau *décisionnel* est ajouté. Ce niveau pourra utiliser les techniques de l'intelligence artificielle pour analyser la mission, la décomposer en tâches exécutables, et décider des actions à accomplir en fonction de l'environnement sans intervention.

Vers la route automatique Des repères seront nécessaires pour passer à la version 3 de PRAXITÈLE : la route automatique. Deux solutions sont en cours d'étude. La première utilise un capteur magnétique détectant des plots posés sur la route. La deuxième utilise la caméra linéaire (utilisée dans le train de véhicule) munie d'un flash éclairant des cibles posées sur les bords des routes.

Portabilité des applications

Il est souhaitable d'améliorer la portabilité des mêmes applications d'une architecture vers une autre indépendamment d'OS temps réel.

Lien vers d'autres outils

– Le niveau de spécification des TRs ou des lois de commande est naturellement accessible par l'automaticien. L'ajout d'une interface, de spécification des lois de commandes d'une manière proche de sa culture - une spécification type schéma-bloc ou d'un ensemble d'équations différentielles formelles - peut être un plus. Comme exemple, on peut citer le simulateur des systèmes dynamiques hybrides SCICOS [NS97].

– La simulation des effets de la décomposition en TMs des shémas de commande à l'exécution sur une architecture cible, doit permettre de tester l'influence du "temps réel" sur ces commandes. Le simulateur doit tenir compte aussi des événements discrets qui traduise le comportement des commandes. Des travaux ont été réalisés dans le projet ICARE de l'INRIA Sophia-Antipolis afin d'utiliser le simulateur SIMPARC [AB92] pour valider les programmes ORCCAD, mais il n'existe pas encore de passerelle directe entre les deux systèmes.

– La génération du code répartie sur une architecture mutli-processeurs en minimisant la période d'asservissement. Une liaison avec le système SYNDEX [Sor96] permet d'optimiser les ressources physiques et de produire un code indépendant de l'architecture matérielle.

Troisième partie

Annexes

Annexe A

Calcul au Point de Contact

A.1 Notation et repères

La modélisation nécessite l'utilisation de plusieurs repères et de plusieurs matrices de rotation. Il est nécessaire d'établir une convention sur les notations pour faciliter l'écriture des équations. Un repère ayant pour origine O_i et pour base les vecteurs $\{x_i, y_i, z_i\}$ sera noté R_i.

Formule de changement de repère Soient $R_i(O_i, x_i, y_i, z_i)$ et $R_j(O_j, x_j, y_j, z_j)$ deux repères. On note R_i^j la matrice 3×3 où les colonnes représentent respectivement les coordonnées x_i, y_i, z_i exprimées dans la base x_j, y_j, z_j.

La matrice de rotation R_i^j, satisfait les relations suivantes :

$$\left(R_i^j\right)^{-1} = R_j^i = \left(R_i^j\right)^T \tag{A.1-1}$$

La matrice homogène de dimension 4×4 est définie par :

$$\bar{R}_i^j = \begin{bmatrix} R_i^j & [O_iO_j] \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{A.1-2}$$

La matrice homogène satisfait les relations suivantes

$$\left(\bar{R}_i^j\right)^{-1} = \bar{R}_j^i = \begin{bmatrix} \left(R_i^j\right)^T & -\left(R_i^j\right)^T [O_iO_j] \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{A.1-3}$$

Soit $R_k(O_k, x_k, y_k, z_k)$ un troisième repère, les relations suivantes sont des relations très utiles :

$$\bar{R}_i^k = \bar{R}_i^j \bar{R}_j^k \tag{A.1-4}$$

$$R_i^k = R_i^j R_j^k \tag{A.1-5}$$

Composition des vitesses Soient trois repères R_0, R_1 et R_2. R_2 se déplace par rapport à R_1avec les vitesses angulaire et linéaire $\left(\omega_{2/1}, v_{2/1}\right)$ et R_1 se déplace par rapport à R_0 avec les vitesses $\left(\omega_{1/0}, v_{1/0}\right)$. Calculons les vitesses du repère R_2 par rapport au repère R_0. Par différenciation de l'équation (A.1-5), nous aurons :

$$\overrightarrow{\omega_{2/0}} = \overrightarrow{\omega_{1/0}} + \overrightarrow{\omega_{2/1}} \tag{A.1-6}$$

$$\overrightarrow{v_{2/0}} = \overrightarrow{v_{1/0}} + \overrightarrow{v_{2/1}} + \overrightarrow{\omega_{1/0}} \wedge \overrightarrow{O_1O_2} \tag{A.1-7}$$

Considérons maintenant un repère mobile $R_i(O_i, x_i, y_i, z_i)$ se déplaçant avec les vitesses $\left(\omega_{i/0}, V_{i/0}\right)$ et soit un point M lié à R_i (c'est à dire que $d[O_iM]^i/dt = 0$). D'après la formule précédente nous avons :

$$\overrightarrow{V(M)} = \overrightarrow{V_{i/O}} + \overrightarrow{\omega_{i/O}} \wedge \overrightarrow{O_iM}$$

Changement de repères Supposons que l'on veut passer du repère de référence R_0 vers un nouveau repère de référence R_1. Si $\bar{R}_i^0$ représente la position du repère R_i exprimée dans le repère R_0 alors $\bar{R}_i^1 = \bar{R}_0^1 \bar{R}_i^0$ représente la position du repère R_i exprimée dans le repère R_1.

Si le repère R_0 se déplace à des vitesses $(\overrightarrow{v_{0/0}}, \overrightarrow{\omega_{0/0}})$, les nouvelles expressions de ces vitesses $(\overrightarrow{v_{1/0}}, \overrightarrow{\omega_{1/0}})$ exprimées dans le repère R_1 s'écrivent :

$$\begin{aligned} \overrightarrow{\omega_{1/0}} &= R_0^1 \overrightarrow{\omega_{0/0}} \\ \overrightarrow{v_{1/0}} &= R_0^1 \overrightarrow{v_{0/0}} + R_0^1(\overrightarrow{\omega_0} \wedge [\overrightarrow{O_0O_1}]_0) \end{aligned}$$

A.2 Calcul de la vitesse au point de contact C

Pour calculer les vitesses au point de contact C par rapport au repère fixe R_0, en plus du repère mobile $R_1(P, \vec{i}_1, \vec{j}_1)$ associé à la plate-forme trois autre repères sont utilisés :

- $R_A(A, \vec{i}_A)$ lié au levier AO_r,
- $R_{O_r}(O_r, \vec{i}_{Or})$ lié à la roue,
- $R_r(O_r, \vec{i}_r, \vec{j}_r\ )$ ayant le même centre que R_{Or} mais attaché au levier O_rA.

La roue est entièrement définie par :

- l'angle $\alpha \equiv (\vec{i}_1, \overrightarrow{PA})$ et $l = \left\|\overrightarrow{PA}\right\|$ pour décrire la position du point de rotation A dans le repère R_1,

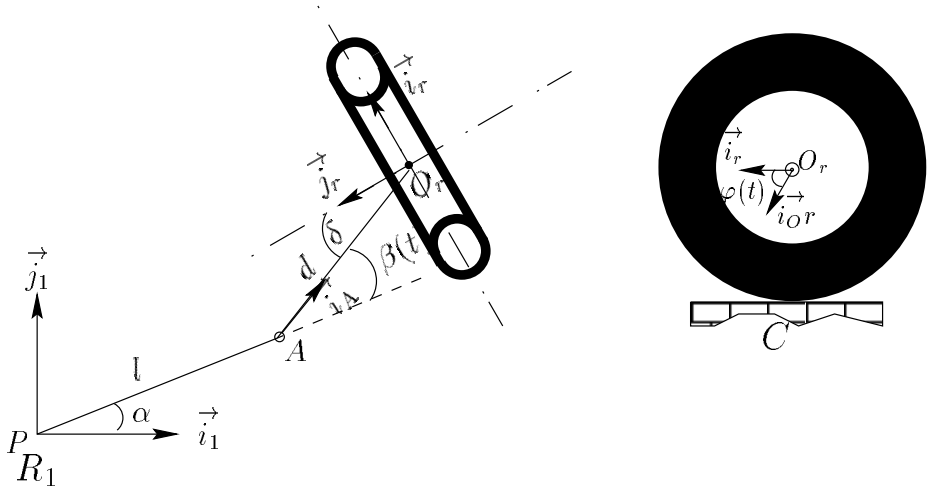


FIG. A.2-1 – *Description d'une roue.*

– l'angle $\beta \equiv (\vec{P}A, \vec{i}_A)$ et $d = \left\|\overrightarrow{O_r A}\right\|$pour décrire la position et l'orientation de la tige $O_r A$,

– l'angle $\delta \equiv (\vec{j}_r, -\vec{i}_A)$ donnant l'orientation de la tige par rapport à la roue,

– l'angle φ donnant l'orientation da la roue autour de son axe de rotation,

– le rayon de la roue r.

En utilisant la loi de composition des vitesses :

$$\overrightarrow{V_{C/O}} = \overrightarrow{V_{P/O}} + \overrightarrow{V_{C/P}} + \overrightarrow{\omega_{P/O}} \wedge \overrightarrow{PC} \quad \text{(A.2-8)}$$

$$\overrightarrow{V_{C/P}} = \overrightarrow{V_{A/P}} + \overrightarrow{V_{C/A}} + \overrightarrow{\omega_{A/P}} \wedge \overrightarrow{AC} \quad \text{(A.2-9)}$$

$$\overrightarrow{V_{C/A}} = \overrightarrow{V_{O_r/A}} + \overrightarrow{V_{C/O_r}} + \overrightarrow{\omega_{O_r/A}} \wedge \overrightarrow{O_r C} \quad \text{(A.2-10)}$$

Sachant que les couples des points P et A, A et O_r, O_r et C appartiennent aux même corps, les vitesses relatives $\overrightarrow{V_{A/P}}$, $\overrightarrow{V_{O_r/A}}$ et $\overrightarrow{V_{C/O_r}}$ sont nulles. Les équations (A.2-8) (A.2-9) (A.2-10) donnent :

$$\overrightarrow{V_{C/O}} = \overrightarrow{V_{P/O}} + \underbrace{\underbrace{\overrightarrow{\omega_{O_r/A}} \wedge \overrightarrow{O_r C}}_{\overrightarrow{V_{C/A}}} + \overrightarrow{\omega_{A/P}} \wedge \overrightarrow{AC}}_{\overrightarrow{V_{C/P}}} + \overrightarrow{\omega_{P/O}} \wedge \overrightarrow{PC} \quad \text{(A.2-11)}$$

Ou bien :

$$\overrightarrow{V_{C/O}} = \overrightarrow{V_{P/O}} + \overrightarrow{\omega_{O_r/A}} \wedge \overrightarrow{O_r C} + \overrightarrow{\omega_{A/P}} \wedge \underbrace{\left(\overrightarrow{AO_r} + \overrightarrow{O_r C}\right)}_{\overrightarrow{AC}} + \overrightarrow{\omega_{P/O}} \wedge \underbrace{\left(\overrightarrow{PA} + \overrightarrow{AO_r} + \overrightarrow{O_r C}\right)}_{\overrightarrow{PC}}$$

$$= \overrightarrow{V_{P/O}} + \underbrace{\left(\overrightarrow{\omega_{O_r/A}} + \overrightarrow{\omega_{A/P}} + \overrightarrow{\omega_{P/O}}\right)}_{\overrightarrow{\omega_{O_r/O}}} \wedge \overrightarrow{O_rC} + \underbrace{\left(\overrightarrow{\omega_{A/P}} + \overrightarrow{\omega_{P/O}}\right)}_{\overrightarrow{\omega_{A/O}}} \wedge \overrightarrow{AO_r} + \overrightarrow{\omega_{P/O}} \wedge \overrightarrow{PA}$$

Finalement :

$$\overrightarrow{V_{C/O}} = \overrightarrow{V_{P/O}} + \overrightarrow{\omega_{P/O}} \wedge \overrightarrow{PA} + \overrightarrow{\omega_{A/O}} \wedge \overrightarrow{AO_r} + \overrightarrow{\omega_{O_r/A}} \wedge \overrightarrow{O_rC} \qquad \text{(A.2-12)}$$

En projetant la vitesse donnée par l'équation (A.2-12) dans le repère R_r :

$$\begin{aligned}
\overrightarrow{V^r_{C/O}} &= R^r_1 R^1_0 \overrightarrow{V^0_{P/O}} + R^r_1 \overrightarrow{\omega^1_{P/O}} \wedge R^r_1 \overrightarrow{PA^1} \\
&+ \left(R^r_1 \overrightarrow{\omega^1_{P/O}} + R^r_A \overrightarrow{\omega^A_{A/1}}\right) \wedge R^r_A \overrightarrow{AO^A_r} \\
&+ \left(R^r_1 \overrightarrow{\omega^1_{P/O}} + R^r_A \overrightarrow{\omega^A_{A/1}} + R^r_{O_r} \overrightarrow{\omega^{O_r}_{O_r/r}}\right) \wedge \overrightarrow{O_rC^r}
\end{aligned}$$

R^1_0 : la matrice de passage du repère R_0 au repère R_1.

$$R^1_0 = \left(R^0_1\right)^T = rot(\vec{k}, \theta)^T = R^T(\theta)$$

R^r_1 : la matrice de passage du repère R_1au repère R_r.

$$\begin{aligned}
R^r_1 &= rot(\vec{k}, -(\alpha + \beta - \delta + \pi/2)) \\
&= \begin{bmatrix} -\sin(\alpha+\beta-\delta) & \cos(\alpha+\beta-\delta) & 0 \\ -\cos(\alpha+\beta-\delta) & -\sin(\alpha+\beta-\delta) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

R^r_A : la matrice de passage du repère R_A au repère R_r.

$$\begin{aligned}
R^r_1 &= rot(\vec{k}, -(-\delta + \pi/2)) \\
&= \begin{bmatrix} \sin\delta & \cos\delta & 0 \\ -\cos\delta & \sin\delta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

R^r_{Or} : la matrice de passage du repère R_{Or} au repère R_r.

$$\begin{aligned}
R^r_1 &= rot(\vec{j}, \varphi) \\
&= \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix}
\end{aligned}$$

$$\omega^1_{P/O} = \omega^1_{1/O} = \begin{pmatrix} 0 \\ 0 \\ \dot{\theta} \end{pmatrix} \omega^1_{A/O} = \begin{pmatrix} 0 \\ 0 \\ \dot{\beta} \end{pmatrix} \omega^{Or}_{Or/r} = \begin{pmatrix} 0 \\ \dot{\varphi} \\ 0 \end{pmatrix}$$

et

$$PA^1 = \begin{pmatrix} l\cos\alpha \\ l\sin\alpha \\ 0 \end{pmatrix} AO_r^A = \begin{pmatrix} d \\ 0 \\ 0 \end{pmatrix} O_rC^r = \begin{pmatrix} 0 \\ 0 \\ -r \end{pmatrix}$$

Enfin, la vitesse au point de contact C de la roue avec le sol exprimée dans le repère R_{O_r} est donnée par :

$$V_{C/O}^r = \begin{bmatrix} -\sin(\alpha+\beta-\delta) & \cos(\alpha+\beta-\delta) & l\cos(\beta-\delta)+d\cos\delta \\ -\cos(\alpha+\beta-\delta) & -\sin(\alpha+\beta-\delta) & l\sin(\delta-\beta)+d\sin\delta \end{bmatrix} R^T(\theta)\dot{\xi} + \begin{bmatrix} d\dot{\beta}\cos\delta - r\dot{\varphi} \\ d\dot{\beta}\sin\delta \end{bmatrix} \quad \text{(A.2-13)}$$

Annexe B

Filtrage des Mesures et Estimation

B.1 Estimation

L'implantation de la commande (longitudinale ou latérale) n'est possible que si toutes les variables d'état du système (position,vitesse, accélération,...) sont physiquement accessibles. En pratique, ces informations fournies par les capteurs forment un vecteur de mesure y, de dimension inférieure à celle de x : y s'exprime lui-même linéairement en fonction de x (et éventuellement de u) sous la forme $y = Cx$.

Pour implanter la commande, il convient donc de procéder à une reconstruction de l'état x à partir des seules informations dont on dispose, à savoir les mesures y. La matrice C n'étant pas inversible, la reconstruction ne peut être immédiate. Cependant on peut obtenir un résultat satisfaisant en utilisant un estimateur (appelé aussi observateur, reconstructeur ou filtre), qui fait une reconstruction approchée $\hat{x}$ de l'état x.

L'implantation de la commande est alors possible en remplaçant x par $\hat{x}$.

B.1.1 En continu

Nous considérons le cas où le système n'est pas perturbé par des bruits : c'est à dire que le système évolue dans un environnement *déterministe*. Dans ce cas , le système s'écrit sous la forme :

$$\begin{cases} \dot{x}(t) = A(t)x(t) \\ y(t) = C(t)x(t) \end{cases}$$

Definition B.1.1 *On appelle observateur du système un opérateur qui génère une approximation $\hat{z}(t)$ de la variable $z(t) = T(t)x(t)$ sous la forme dynamique :*

$$\dot{\hat{z}}\,(t) = F\,(t)\,\hat{z}(t) + K^{\circ}\,(t)\,y(t) \qquad \text{(B.1-1)}$$

Si z et x sont de même dimension, l'observateur est dit "complet". On prend toujours alors $z = x$ et $\hat{z} = \hat{x}$, soit $T = I_n$ (matrice identité) auquel cas on parle d'observateur identité.

Dans un contexte *déterministe*, les qualités à exiger d'un observateur sont les suivants :

– il doit être stable,

– il doit assurer la convergence de $\hat{z}$ vers z, de manière exacte dans un contexte déterministe.

Ces deux points amènent la définition et le théorème suivant :

Definition B.1.2 *Dans le cas déterministe, un observateur est dit sans biais si :*

$$quel\ que\ soit\ x(t_0)\ pour\ t \geq t_0$$

$$\begin{cases} \hat{z}(t_0) = z(t_0) \Longrightarrow \hat{z}(t) = z(t) \\ \hat{z}(t_0) \neq z(t_0) \Longrightarrow \lim_{t\to\infty} [\hat{z}(t) - z(t)] = 0 \end{cases}$$

Théorème B.1.3 *Un observateur est sans biais si et seulement si :*

– $F(t)$ *est exponentiellement stable;*

– *et*

$$T(t)A(t) + \dot{T}(t) - F(t)T(t) = K(t)C(t) \tag{B.1-2}$$

– *et*

$$D(t) = T(t)B(t) \tag{B.1-3}$$

Le théoreme est démontré dans [DUC87].

Particularité de l'estimateur

Notre observateur est "complet" donc : $T = I$; $z = x$ et $\hat{z} = \hat{x}$.

Les équations (B.1-2) et (B.1-3) du théorème deviennent :

$$F(t) = A(t) - K(t)C(t) \tag{B.1-4}$$

$$D(t) = B(t) \tag{B.1-5}$$

donc l'observateur (B.1-1) s'écrira :

$$\dot{\hat{x}}(t) = A(t)\hat{x}(t) + K^c(t)[y(t) - C(t)\hat{x}(t)]$$

Si :

$$\hat{x}(t) = \begin{pmatrix} \hat{x}_1(t) \\ \hat{x}_2(t) \end{pmatrix}$$

① La mesure est toujours effectuée sur $\hat{x}_1$, c'est à dire :

$$C(t) = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

② $\hat{x}_2$ est la dérivée de $\hat{x}_1$:

$$A(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

③ Les gains de la matrices $K^c(t)$ sont constants :

$$K^c(t) = \begin{pmatrix} K_1^c \\ K_2^c \end{pmatrix}$$

L'estimateur en continu s'écrira :

$$\dot{\hat{x}}(t) = [A(t) - K^c(t)C(t)]\hat{x}(t) + K^c(t)y(t)$$

soit après une transformée de Laplace :

$$\hat{X}(s) = [sI - (A(t) - K_t C(t))]^{-1} K^c(t) Y(s)$$

$$\hat{X}(s) = \frac{1}{K_2^c + K_1^c s + s^2} \begin{pmatrix} K_2^c + K_1^c s \\ K_2^c s \end{pmatrix}$$

Si on considère la deuxième variable d'état, on a :

$$\frac{\hat{X}_2(s)}{Y(s)} = \frac{K_2^c s}{K_2^c + K_1^c s + s^2}$$

or $\dot{\hat{x}}_1 = \hat{x}_2$ donc : $s\hat{X}_1(s) = \hat{X}_2(s)$
d'ou :

$$\frac{\hat{X}_1(s)}{Y(s)} = \frac{K_2^c}{K_2^c + K_1^c s + s^2} \tag{B.1-6}$$

Donc le filtrage de la mesure Y par à un filtre de deuxième ordre produit une estimation de la première composante de X. Si l'on considère que ce filtre a une pulsation propre ω_0 et un amortissement ξ, alors :

$$\begin{aligned} K_2^c &= \omega_0^2 \\ K_1^c &= 2\xi\omega_0 \end{aligned}$$

B.1.2 En discret

L'observateur dont on cherche à reconstruire l'état s'écrit :

$$\begin{cases} x_{k+1} = A_k x_k \\ y_k = C_k x_k \end{cases}$$

En transportant directement les résultats précédents, on obtient les définitions et théorème suivants :

Definition B.1.4 *On appelle observateur du système un opérateur qui génère une approximation $\hat{z}_k$ de $z_k = T_k x_k$ sous la forme :*

$$\hat{z}_{k+1} = F_k \hat{z}_k + K_k^d y_k \tag{B.1-7}$$

Definition B.1.5 *Dans le cas déterministe, l'observateur est dit sans biais si :*

$quel\ que\ soit\ x_{k_0}\ pour\ k \geq k_0$

$$\begin{cases} \hat{z}_{k_0} = z_{k_0} \Longrightarrow \hat{z}_k = z_k \\ \hat{z}_{k_0} \neq z_{k_0} \Longrightarrow \lim_{k\to\infty} [\hat{z}_k - z_k] = 0 \end{cases}$$

Théorème B.1.6 *Un observateur est sans biais si et seulement si :*

– F_k *est exponentiellement stable;*

– *et*

$$T_{k+1} A_k - F_k T_k = K_k^d C_k \tag{B.1-8}$$

– *et*

$$D_k = T_{k+1} B_k \tag{B.1-9}$$

Le théoreme est démontré dans [DUC87].

Particularité de l'estimateur

Notre observateur est "complet", les équations (B.1-8) et (B.1-9) du théorème deviennent :

$$F_k = A_k - K_k^d C_k \tag{B.1-10}$$

$$D_k = B_k \tag{B.1-11}$$

L'observateur (B.1-7) s'écrira :

$$\begin{aligned} \hat{x}_{k+1} &= \left(A_k - K_k^d C_k\right) \hat{x}_k + K_k^d y_k \\ &= A_k \hat{x}_k + K_k^d \left[y_k - C_k \hat{x}_k\right] \end{aligned}$$

sachant que :

$$\begin{aligned} A_k &= \exp(A(t)T_e) \\ &= \begin{pmatrix} 1 & T_e \\ 0 & 1 \end{pmatrix} \end{aligned}$$

$$C_k = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

$$K_t^d = \begin{pmatrix} K_1^d \\ K_2^d \end{pmatrix}$$

$$\hat{x}_{k+1} = \begin{pmatrix} 1 - K_1^d & T_e \\ -K_2 & 1 \end{pmatrix} + K_k^d y_k \quad \text{(B.1-12)}$$

$$= F_k \hat{x}_k + K_k^d y_k \quad \text{(B.1-13)}$$

en transformée en z :

$$z\hat{X}_k = F_k \hat{X}_k + K_k^d Y(z)$$

ou bien :

$$\hat{X}_k = \frac{1}{1 - K_1^d + K_2^d T_e + (K_1^d - 2)z + z^2} \begin{pmatrix} K_1^d z - K_1^d + K_2^d T_e \\ K_2^d (z-1) \end{pmatrix}$$

Si on considère :

$$\frac{\hat{X}_{1,k}(z)}{Y(z)} = \frac{K_1^d z - K_1^d + K_2^d T_e}{1 - K_1^d + K_2^d T_e + (K_1^d - 2)z + z^2}$$

$$\frac{\hat{X}_{2,k}(z)}{Y(z)} = \frac{K_2^d (z-1)}{1 - K_1^d + K_2^d T_e + (K_1^d - 2)z + z^2}$$

or $\dot{\hat{x}}_1 = \hat{x}_2$ donc : $\frac{z-1}{z}\hat{X}_{1,k}(z) = \hat{X}_{2,k}(z)$
ce qui fait que :

$$\frac{\hat{X}_{1,k}(z)}{Y(z)} = \frac{K_2^d z^{-1}}{(1 - K_1^d + K_2^d T_e)z^{-2} + (K_1^d - 2)z^{-1} + 1} \quad \text{(B.1-14)}$$

B.1.3 Relation entre les gains discrets et les gains continus

Soit :

$$\frac{\hat{X}_1(s)}{Y(s)} = \frac{K_2^c}{K_2^c + K_1^c s + s^2} = \frac{1}{(s + a + jb)(s + a - jb)} \quad \text{(B.1-15)}$$

avec pour $\xi < 1$
$a = \xi\omega_0$
$b = \omega_0\sqrt{1 - \xi^2}$

Les pôles $s_1 = (-a - jb)$ et $s_2 = (-a + jb)$ de B.1-15 se transforment dans le plan Z en pôles suivants : $z_1 = \exp((-a - jb)T_e)$ et $z_2 = \exp((-a + jb)T_e)$. D'autre part, cette fonction de transfert n'a pas de zéros finis dans le plan S, mais elles a deux zéros infinis.

Ces zéros s'écrivent dans le plan Z : $z = -1$. Le filtre équivalent en discret peut être donné par :

$$\frac{\hat{X}_1(z)}{Y(z)} = K \frac{(z+1)^2}{z^2 - 2z\exp(-aT_e)\cos bT_e + \exp(-2aT_e)}$$

le gain statique doit satisfaire la condition :

$$\frac{\hat{X}_1(s=0)}{Y(s=0)} = \frac{\hat{X}_1(z=1)}{Y(z=1)}$$

ce qui donne :

$$K = \frac{1 - 2\exp(-aT_e)\cos bT_e + \exp(-2aT_e)}{4(a^2+b^2)}$$

finalement :

$$\frac{\hat{X}_1(z)}{Y(z)} = \frac{1 - 2\exp(-aT_e)\cos bT_e + \exp(-2aT_e)}{4(a^2+b^2)} \frac{(1+z^{-1})^2}{1 - 2\exp(-aT_e)z^{-1}\cos bT_e + \exp(-2aT_e)z^{-2}}$$

En égalisant le dénominateur de cette équation avec l'équation B.1-14, on obtient :

$$K_1^d = -2\exp(-aT_e)\cos bT_e + 2$$

$$K_2^d = \frac{1}{T_e}\{\exp(-2aT_e) - 1 + K_1^d\}$$

L'amortissement est choisi de telle manière à avoir un amortissement optimal c'est à dire $\xi = \frac{1}{\sqrt{2}}$. On choisira une pulsation propre ω_0 plus petite pour les axes qui tournent vites.

B.2 Filtre sous forme d'équations récurrentes

La programmation du filtrage sera faite d'une façon récurrente. L'algorithme est constitué de deux étapes principales :

① prédiction du vecteur d'état grâce à la dynamique modélisée dans l'équation d'état par :

$$\hat{x}_{k+1/k} = A_k \hat{x}_{k/k}$$

où A_k est le modèle donné par la matrice, ce qui donne, en décomposant les matrices :

$$\hat{x}_{1,k+1/k} = \hat{x}_{1,k/k} + \hat{x}_{2,k+1/k} T_e$$

$$\hat{x}_{2,k+1/k} = \hat{x}_{2,k/k}$$

② mise à jour de l'état par estimation à partir de la dernière mesure arrivée. L'innovation de la mesure, qui est la différence entre la valeur mesurée à cet instant y^m_{k+1}et la valeur déduite de la prédiction, soit $\hat{y}_{k+1} = C_{k+1}\hat{x}_{k+1/k}$ permet d'obtenir à partir de $\hat{x}_{k+1/k}$une nouvelle estimation de $\hat{x}_{k+1/k+1}$au moyen d'un terme correctif K^d.

$$y_{innov} = (y_{k+1} - \hat{y}_{k+1}) \quad \text{(B.2-16)}$$

$$= (y_{k+1} - C_{k+1}\hat{x}_{k+1/k}) \quad \text{(B.2-17)}$$

$$\hat{x}_{k+1/k+1} = \hat{x}_{k+1/k} + K^d y_{innov} \quad \text{(B.2-18)}$$

En décomposant :

$$y_{innov} = (y^m_{k+1} - \hat{x}_{1,k+1/k})$$

$$\hat{x}_{1,k+1/k+1} = \hat{x}_{1,k+1/k} + K^d_1 y_{innov}$$

$$\hat{x}_{2,k+1/k+1} = \hat{x}_{2,k+1/k} + K^d_2 y_{innov}$$

Annexe C

La loi de Commande Longitudinale

C.1 Contraintes

La commande longitudinale pour le train de véhicule a été élaborée suivant les contraintes particulières suivantes :

Caractéristiques des véhicules Pour que les véhicules puissent se suivre en train, ils doivent avoir des caractéristiques en vitesse et en accélération sensiblement identiques.

Circulation dans les villes La circulation en ville nécessite la limitation de la vitesse à $50km.h^{-1}$. D'autre part, même si les véhicules sont vides, un déplacement souple nécessite la limitation de l'accélération à $2m.s^{-2}$ et de la décélération à $-5m.s^{-2}$ dans les situations urgentes;

Constante de temps A cause des problèmes d'amplification dans le train, il est impossible de maintenir une distance constante entre les véhicules sans communication (technique qu'on veut éviter) de la vitesse du premier véhicule aux autres véhicules du train.

Pour cela nous avons choisi d'imposer une distance proportionnelle à la vitesse :

$$X_1 - X_2 = d_{\min} + hV_2 \tag{C.1-1}$$

où $-$ X_1 et V_1 (resp. X_2 et V_2) la position et la vitesse du véhicule suivi (resp. du véhicule suiveur);

$-$ $d_{\min}$ représente l'inter-distance entre deux véhicules à l'arrêt;

$-$ h le temps qui sépare deux véhicules.

En dérivant l'équation (C.1-1), on aura :

$$V_1 - V_2 = hA_2 \tag{C.1-2}$$

où

$-$ A_2 représente l'accélération du véhicule suiveur.

En réécrivant l'équation (C.1-2) avec Laplace, on aura :

$$V_2 = \frac{V_1}{1 + hp} \tag{C.1-3}$$

La formule C.1-3 signifie que la vitesse du véhicule suiveur V_2 est obtenue en filtrant la vitesse du véhicule suivi V_1 par un filtre passe bas de pulsation de coupure égale à h^{-1}. Le choix de cette pulsation doit être compatible avec la bande passante du véhicule. Sachant que nous utilisons des véhicules électriques, la constante de temps est directement liée au moteur électrique de locomotion. Elle vaut environ $0.1s$, ce qui impose de choisir $h \geq 0.1\,\mathrm{sec}$.

Spécifications du train Le cahier des charges impose une inter-distance de $0.5\ m$ à l'arrêt et $3\ m$ au maximum à la vitesse autorisée en ville (i.e $50\ km.h^{-1}$). Ceci impose de choisir h au moins égal à $0.2\,\mathrm{sec}$.

C.1.1 Correcteur linéaire à gains constants

L'accélération est prise comme variable de commande. Elle peut varier rapidement entre $-A_{\min}^{long}$ et $+A_{\max}^{long}$. La forme du correcteur qui maintiendrait $dx = hV_2 + d_{\min}$ s'écrira :

$$A_d = C_v dv + C_p(dx - hV_2 - d_{\min}) \tag{C.1-4}$$

avec $dx = X_1 - X_2$ et $dv = V_1 - V_2$; et C_p et C_v respectivement le gain sur la distance et le gain sur la vitesse;

La transmittance s'écrira :

$$\frac{X_2(p)}{X_1(p)} = \frac{C_v p + C_p}{p^2 + (C_v + hC_p)p + C_p} \tag{C.1-5}$$

Or, d'après C.1-3 on veut avoir :

$$\frac{X_2(p)}{X_1(p)} = \frac{1}{1 + hp} \tag{C.1-6}$$

On égalise les équations (C.1-5) et (C.1-6), on aura :

$$\begin{aligned} p^2 + (C_v + C_p)p + C_p &= (C_v p + C_p)(1 + hp) \\ &= hC_v p^2 + (C_v + hC_p)p + C_p \end{aligned}$$

Ce qui impose : $hC_v = 1$ et $(C_v + hC_p) = C_v + C_p h$ ou bien : $C_v = h^{-1}$ et $C_p \leq h^{-2}$

Le choix de $C_p = h^{-2}$ est optimal si la saturation n'est pas prise en compte. Mais comme on a une contrainte sur l'accélération maximale autorisée, les gains C_p et C_v ne seront pas constants. C'est ce que l'on va détailler dans le paragraphe suivant.

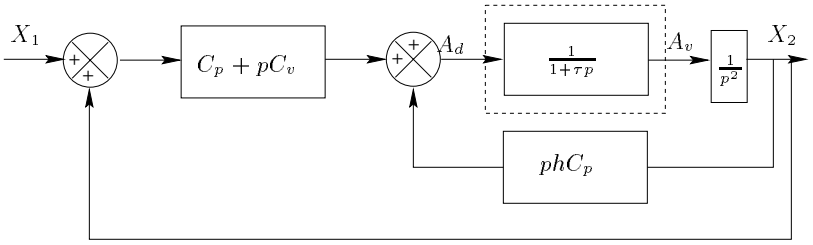


FIG. C.1-1 – *Loi de commande longitudinale avec inter-distance variable.*

C.1.2 Correcteur linéaire à gains variables

On peut définir une zone de sécurité ΔX par :

$$\Delta X \succeq D\{V_2(t), A_2(t)\} - D\{V_1(t), A_1(t)\} \tag{C.1-7}$$

avec :

$D\{V_2(t), A_2(t)\}$ (resp. $D\{V_1(t), A_1(t)\}$) la distance parcourue par le véhicule suiveur (resp. le véhicule suivi) à l'instant t.

Si on prend l'instant particuliér où le véhicule suiveur a atteint sa limite en accélération (i.e $A_{\max}^{long}$) et sa vitesse vaut V_0, alors la distance minimale d'arrêt vaut :

$$D\{V_0, A_{\max}^{long}\} = \frac{V_0^2}{2A_{\max}^{long}} \tag{C.1-8}$$

Une commande optimale s'obtient lorsque :

$$\Delta X = D\{V_2(t), A_2(t)\} - D\{V_1(t), A_1(t)\} \tag{C.1-9}$$

en linéarisant :

$$\Delta X + \frac{\partial D\{V_2(t), A_2(t)\}}{\partial V_2(t)} \Delta V = 0 \tag{C.1-10}$$

On retrouve, le correcteur de la forme :

$$\Delta V + C_p \Delta X = 0 \tag{C.1-11}$$

avec :

$$\begin{aligned} C_p &= \frac{1}{\frac{\partial D\{V_0, A_{\max}^{long}\}}{\partial V_0}} \\ &= \frac{A_{\max}^{long}}{V_2} \end{aligned}$$

En s'inspirant de cette courbe de sécurité, on peut définir un correcteur linéaire à coefficients variables asymptotiquement stable et surtout sécuritaire qu'elles que soientt les conditions initiales :

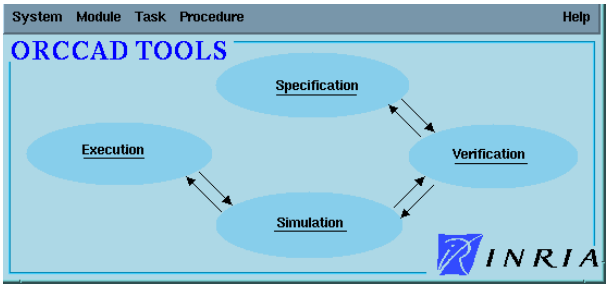
$$A_d = h^{-1}(dv + C_p(dx - hV_2 - d_{\min})) \tag{C.1-12}$$

avec :

- si l'accélération demandée est suffisante pour données à la voiture la vitesse demandée (c'est à dire si $V_2 \leq hA_{\max}$) ,le coefficient proportionnel sera : $C_p = h^{-1}$
- si non, il faut baisser le gain en position en fonction de la vitesse : $C_p = \frac{A_{\max}}{V_2}$.

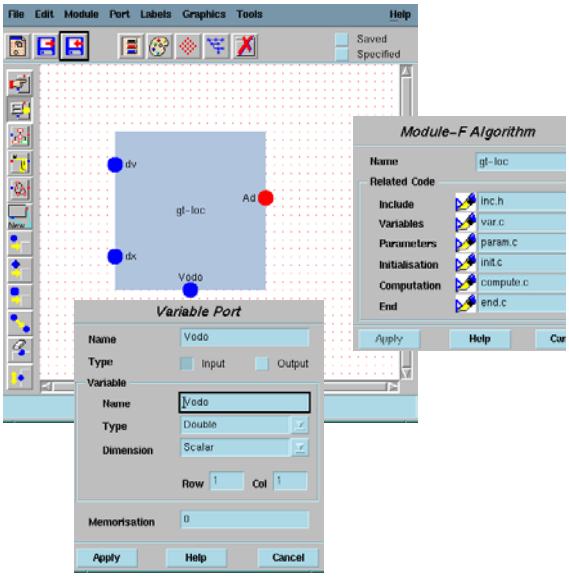
Annexe D

L'interface Homme-Machine d'ORCCAD



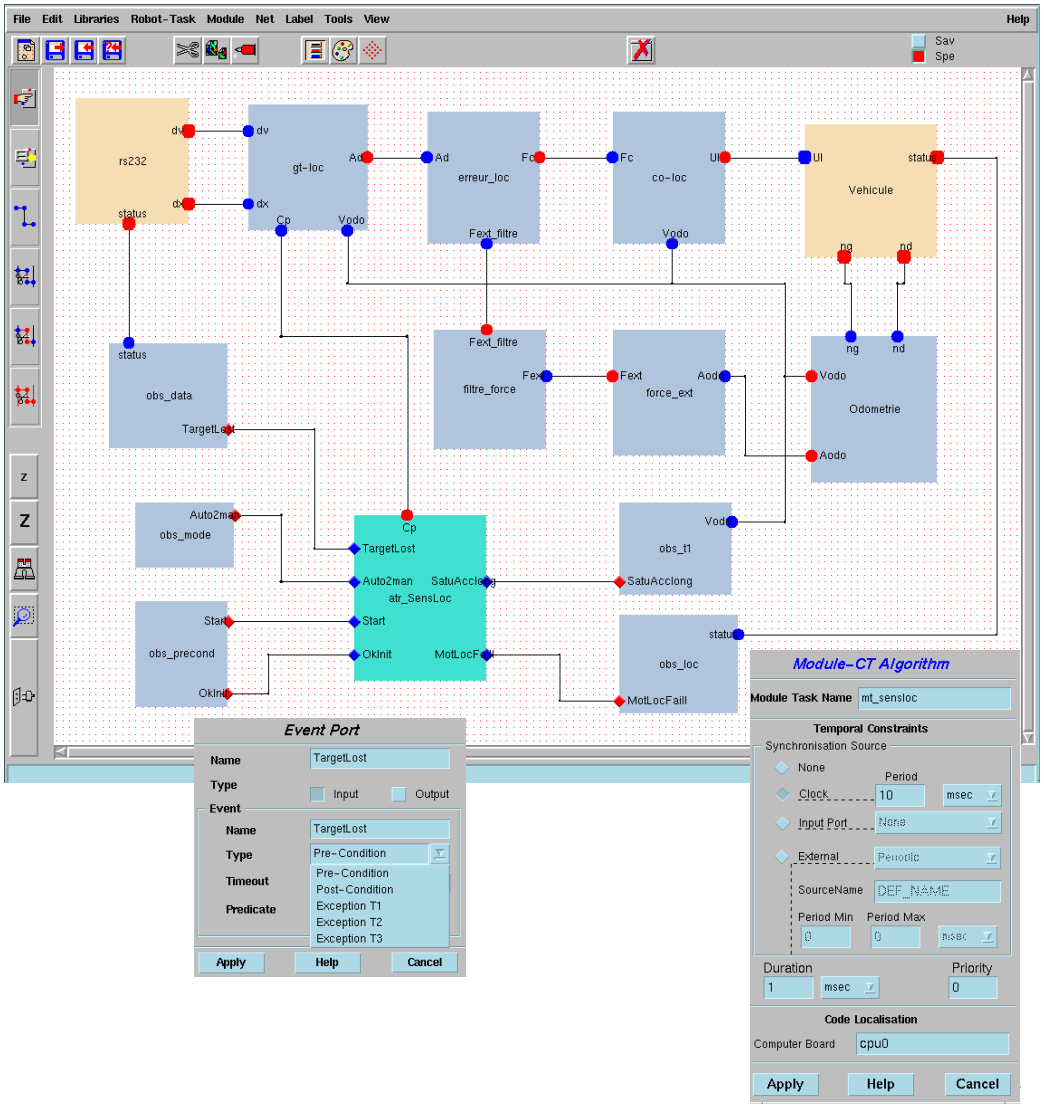
Une nouvelle version d'ORCCAD est en cours de réalisation par collaboration entre le projet ICARE de l'INRIA Sophia-Antipolis et le projet BIP de l'INRIA Rhône-Alpes [SEKPG97, BCME+98]. Cette version est entièrement écrite en C++ utilisant le générateur d'interface ILOG VIEWS. Trois éditeurs sont réalisés pour accéder au divers couches de programmation.

L'éditeur des TMs



L'éditeur des TMs permet de construire les ports ainsi que les codes à exécuter (à l'initialisation, pendant le cycle de calcul et à la fin) et les fichiers de paramètre et de déclaration des variables. Une librairie de TMs déjà établie suivant les domaines d'applications robotiques (mobile, manipulateur, sous-marin, ...) pourra assister le concepteur.

La figure ci-contre montre la construction de la TM GT-LOC qui permet de calculer l'accélération désirée A_d à partir des données dx, dv et $V_o do$ (équation 3.2-1) du chapitre 3.



L’éditeur des TRs

Un deuxième éditeur permet de spécifier une TR en utilisant les TMs déjà construites. A ce niveau il faut :

- lier les ports des TMs;

- donner les caractéristiques temporelles des TMs (période d’activation, la source de synchronisation, ...);

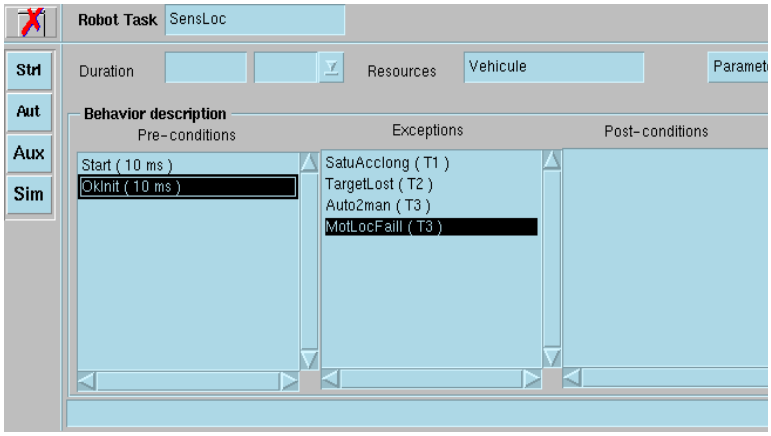
- spécifier le comportement logique de la TR, c’est à dire les événements et les exceptions associées à leurs traitements;

Le code temps réel VxWorks peut être généré automatiquement à partir de cette spécification et pour différentes machines cibles (68XXX, C40, ...).

La figure ci-dessus montre la spécification de la TR SensLoc décrite dans le chapitre 3 de la deuxième partie.

Vue abstraite d'une TR

Une vue abstraite du comportement de la TR permet de générer le code ESTEREL correspondant, de le simuler et de visualiser l'automate correspondant. Des incohérences dans la spécification peuvent être détectées.



Bibliographie

[AADP96] C. Allal, S. Abdou, P. Daviet, and M. Parent. Tactical and technical routing of empty vehicles. In *4th IEEE Mediterranean Symposium on New Directions in Control and Automation*, 1996.

[AB92] C. Astraudo and J.J. Borrelly. Simulation of miltiprocessor robot controllers. In *Proc. IEEE Int. Conf. on Robotics and Automation*, Nice, France, May 1992.

[ABGM96] Ch. André, H. Boufaid, D. Gaffé, and J.P. Marmorat. Environnement pour la programmation synchrone. In *Real-Time Systems and Embedded Systems*, Paris, 1996.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A Henzinger, P.H Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. In *Theorical Computer Science*, pages 3–34, 1995.

[ACS] ACS-Technologies, France. *C-PRS Technical Representation*, 1995 edition.

[ADL+91] M. Alvici, P. Deloof, W. Linss, G. Preti, and A. Rolland. Anticollision radar : state of the art. In *Proc. Advanced Telematics in Road Transport, DRIVE Conference*, volume II, pages 943–961, Brussels, February 1991.

[AEP96a] S. Abdou, B. Espiau, and M. Parent. Mission programming : Application to the distribution of empty vehicles in the praxitele project. In *4th IEEE Mediterranean Symposium on New Directions in Control and Automation*, Chania, Crete, June 1996.

[AEP96b] S. Abdou, B. Espiau, and M. Parent. Specification, formal verification and implementation of tasks and mission for an autonomous vehicle. In *International Symposium on robotics and Manufacturing*, Montpellier, France, June 1996.

[AJRS94] M. Andrews, S. Jacobi, U. Reuter, and V. Salzer. Porche's contribution to safer and more comfortable driving in the future. In *Proc. First World Congress*

on Application of Transport Telematics and Intelligent Vehicle-Highway Systems, pages 1879–1886, Paris, November-December 1994.

[Alb91] J.S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):473–509, 1991.

[Alb95] J.S. Albus. Rcs : A reference model architecture for intelligent systems. In AAAI Press, editor, *In Proceedings of the 1995 AAAI Spring Symposium : Lessons Learned from Implemented Software Architectures for Physical Agents*, March 1995. available via anonymous ftp at hobbes.jsc.nasa.gov/pub/korten/spring_symposium/submissions.

[ALL66] M. Athans, W. Levine, and A. Levis. On the optimal and suboptimal position and velocity control of a string of high-speed moving trains. Technical Report PB 173 640, M.I.T Electronic Systems Laboratory, November 1966.

[AM91] J.M. Adan and M.F. Magalhaes. Developing reconfigurable distributed hard real-time control systems in ster. In *Algorithms and Architectures for Real-Time Control, Proc. of the IFAC Workshop*, pages 147–152, September 1991.

[AML89] J.S. Albus, H.G. McCain, and R. Lumia. Nasa/nbs standard reference model for telerobot control system architecture. Technical report, National Institute of Standart Technology, 1989.

[And72] J.E. Anderson. Personal rapid transit. In *Institut of Technology, University of Minnesota*, 1972.

[And74] J.E. Anderson. Personal rapid transit ii. In *University of Minnesota*, 1974.

[APE95] S. Abdou, M. Parent, and B. Espiau. Spécification et implémentation logicielles pour des véhicules automatiques. In *Real-Time Systems and Embedded Systems*, 1995.

[Ark87] R.C. Arkin. Dynamic replanning for a mobile robot based on internal sensing. In *Proceeding of the IEEE International Conference on Robotics and Automation*, 1987.

[AS90] J. Ackermann and W. Sienel. Robust control for automatic steering. In *Proc. 1990 American Control Conference*, pages 795–800, San Diego, May 1990.

[ASSU94] J. Ackermann, W. Sienel, R. Steinhauser, and V.I. Utkin. Linear and nonlinear controller design for robust automatic steering. *IEEE Trans. on Contr. Sys. Tech.*, 1994.

[AW93] Stove A.G. and Chodyniecki W. Radar sensor for aicc. In *PROMETHEUS Futur Systems, Autotech '93*. Institute of Mechanical Engineering, 1993.

[Bar62] D. Barrick. Automatic steering techniques. In *IRE Inter. Conv. Rec*, volume Part 2, pages 166–178, 1962.

[BB91] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceeding of the IEEE*, 79(9), 1991.

[BBD+94a] S. Becker, M. Bork, H.T. Dorissen, G. Geduld, O. Hofmann, K. Naab, G. N'ocker, P. Rieth, and J. Sonntag. Summary of experience with autonomous intelligent cruise control (aicc) part i : study objectives and methods. In *Proc. First World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 1828–1835, Paris, November-December 1994.

[BBD+94b] S. Becker, M. Bork, H.T. Dorissen, G. Geduld, O. Hofmann, K. Naab, G. N'ocker, P. Rieth, and J. Sonntag. Summary of experience with autonomous intelligent cruise control (aicc) part i : results and conclusions. In *Proc. First World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 1836–1843, Paris, November-December 1994.

[BC58] J.B. Bidwell and R.S. Cataldo. Electronic chauffeur may guide car of future. *SAE J.*, pages 64–67, September 1958.

[BCME+98] J. J. Borrelly, E. Coste-Manière, Bernard Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, and N. Turro. An integrated and modular approach for the specification, the validation and the implementation of complex robotics missions. *International Journal of Robotics Research*, 1998.

[BDR94] M. De Saint Blancard, B. Dubuisson, and M. Rombaut. Pro-lab2 : A vehicle with driving assistance. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle Highway Systems*, pages 1871–1878, November-December 1994.

[BdS91] F. Boussinot and R. de Simone. The ESTEREL language. *Proceeding of the IEEE*, 79:1293–1304, september 1991.

[BDS94] T. Butsuen, A. Doi, and H. Sasaki. Development of a collision avoidance system with automatic brake control. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2079–2086, Paris, November-December 1994.

[Ben81] J.G. Bender. System studies of automated hightway systems. Technical Report EP-81041A, General Motors Transportations Systems Center Report, 1981. FHWA Report No. FHWA/RD-82/003, 1982.

[Ben91] J.G. Bender. An overview of systems studies of automated highway systems. *IEEE Trans. Veh. Tech.*, 40(1):82–99, 1991.

[Ber89] G. Berry. Real time programming : Special purpose languages or general purpose languages. In *the 11th IFIP World Congress*, San Fransisco, 1989.

[Ber95] D. Bertrand. Parking automatique de véhicule electrique par capteurs ultrasonors. Rapport du Projet de fin d'études (E.N.S.M.P), Juin 1995.

[BF70] J. Bender and R. Fenton. On the flow capacity of automated highways. *Trans. Sci.*, 4:52–63, 1970.

[BFK90] H.B. Brown, M.B. Friedman, and T. Kanade. Development of a 5-dof walking robot for space station application : overview. In *IEEE International Conference on Systems Engineering*, pages 194–197, August 1990.

[BFO71] J. Bender, R.E. Fenton, and K. Olson. An experimental study of vehicle automatic longitudinal control. *IEEE Trans. Veh. Tech.*, 4(20):114–123, 1971.

[Bis94] R.A. Bishel. Dual-mode truck : automated and manual operation. In *SAE Future Transportation Technology Conference*, 1994.

[BK94] R.P. Bonasso and D. Kortenkamp. An intelligent agent architecture in which to pursue robot learning. In *Robot Learning Workshop*, July 1994.

[BLG+94] P. Bournai, Ch. Lavarenne, P. Le Guernic, O. Mafféıs, and Y. Sorel. Interface signal-syndex. Rapport de recherche 2206, INRIA, 1994.

[BLMM91] F. Broqua, G. Lener, V. Mauro, and E. Morello. Cooperative driving : basic concepts and a first assessment of 'intelligent cruise control' strategies. In *Proc. Advanced Telematics in Road Trasport*, volume II, pages 908–929, Brussels, Belgium, February 1991.

[Bon74] L.S. Bonderson. Optimal lateral control for dual-mode vehicles. In *Dual-Mode Conference*, Washington, D.C., May 1974.

[Boo86] G. Booch. Object-oriented development. *IEEE Transactions on Software Engineering*, pages 211–221, February 1986.

[BP97] S. Bensoussan and M. Parent. Computer-aided teleoperation of an urban vehicle. In *ICAR'97*, Monterey, CA, July 1997.

[Bro74] S.J. Brown. Design consideration for vehicle state control by the point-follower method. In *Department of Audio Visual Extension, University of Minnesota, MN*, pages 381–390. Personal Rapid Transit II, 1974.

[Bro89] R.A. Brooks. A robot that walks; emergent behavior from a carefully evolved network. In *Proceeding of the IEEE International Conference on Robotics and Automation*, May 1989.

[BS94] J.R. Bishop and W. Stevens. Results of precursor systems analyses of automated highway systems. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 1959–1966, Paris, November-Dcember 1994.

[CADL92] R. Chatila, R. Alami, B. Degallaix, and H. Laruelle. Integrated planning and execution control of autonomous robot actions. In *IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.

[Car70] K.H.F Cardew. the automatic steering of vehicles - an experimental system fitted to a ds citroen car. Technical report, Road Research Laboratory Report RRL LR 340, Crowthorne, England, 1970. (PB 196 983).

[CB80] R.J. Caudill and S.L. Blasnik. Model-reference longitudinal control for automated vehicle systems. *ASME J. Dyn. Sys. Meas. Contr.*, (102), 1980.

[CBAN96] G. Campion, G. Bastin, and B. Andréa-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62, February 1996.

[CDPW] H. Christ, W. Darenberg, F. Panik, and W. Weidemann. Automatic track control of vehicles. In *The Dynamics of Vehicles on Roads and Tracks. Swets and Zeitlinger*, pages 145–164, Amsterdam.

[CES83] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporel-logic specifications : a practical approch. In *ACM symposium on Principles of Programming Languages and Systems*, 1983.

[CG77a] R.J. Caudill and W.L. Garrard. Vehicle follower longitudinal control for automated guideway transit vehicles. Umta-mn-11-0002-77-1, University of Minnesota, February 1977.

[CG77b] R.J. Caudill and W.L. Garrard. Vehicle follower longitudinal control for automated transit vehicles. *J. Dyn. Sys. Meas. Contr.*, 4(99):241–248, 1977.

[CG94] C. Cugiani and L. Giubbolini. Millimetre wave radar sensor for the highway global positioning of a vehicle. In *Proc. 1994 Vehicle Navigation and Information Systems*, pages 409–414, Yokohama, Japan, August-September 1994.

[CH95a] S. Choi and J.K. Hedrick. Comparison of engine models and distance sensors for vehicle longitudinal control : analysis and test results in path. In *ITS America Annual Meeting*, Washington, D.C., March 1995.

[CH95b] S. Choi and J.K. Hedrick. Vehicle longitudinal control using an adaptive observer for automated highway systems. In *Proc. 1995 American Control Conference*, Seattle, WA, June 1995.

[Chi79] H.Y. Chiu. A state-constrained vehicle-follower approach to the station-egress problem. *IEEE Trans. Veh. Tech.*, 1(28):70–79, 1979.

[CHZ+93] K.S. Chang, J.K. Hedrick, W.B. Zhang, P. Varaiya, M. Tomizuka, and S.E. Shladover. Automated highway system experiments in the path program. In *IVHS J*, pages 63–87, 1993.

[CI92] C.C. Chien and P. Ioannou. Automatic vehicle-following. In *Proc. 1992 American Control Conference*, pages 1748–1752, Chicago, IL, June 1992.

[CIL94] C.C. Chien, P.A. Ioannou, and M.C. Lai. Entrainment and vehicle following controllers design for autonomous intelligent vehicles. In *Proc. American Control Conference*, pages 6–10, Baltimore, MD, June 1994.

[CLD+91] K.S. Chang, W. Li, P. Devlin, A. Shaikhbahai, P. Varaiya, J.K. Hedrick, D. McMahon, V. Narendran, D. Swaroop, and J. Olds. Experimentation with a vehicle platoon control system. In *Proc. Vehicle Navigation and Information Systems Conference*, pages 1117–1124, Dearborn, MI, October 1991.

[CMER92] E. Coste-Manière, B. Espiau, and E. Rutter. A task-level robot programming language and its reactive execution. In *Proceeding of the 1992 IEEE International Conference on Robotics and Automation*, pages 2751–2756, Nice, France, 1992.

[CMT82] R.J. Caudill, P. Di Mattco, and S.P. Thomas. Longitudinal control for automated highway vehicles. *J. Dyn. Sys. Meas. Contr.*, 2(104):180–187, 1982.

[CMT97] E. Coste-Manière and N. Turro. The MAESTRO language and its environment: Specification, validation and control of robotic missions. In *IEEE/RSJ Intl Conf on Intelligent Robots and Systems, IROS'97*, pages vol 2, p 836–841, Grenoble, France, September 1997.

[Con92] J.H. Connell. Sss : A hybride architecture applied to robot navigation. In *IEEE International Conference on Robotics and Automation*, Nice, France, May 1992.

[Cor77] Calspan Corp. Practicability of automated highway systems, 1977.

[CPS89] R. Cleaveland, J. Parrow, and B. Steffen. The concurrency workbench. Berlin, june 1989.

[CSB76] H.Y. Chiu, G.B. Stupp, and S.J. Brown. Vehicle-follower controls for short headway agt systems - functional analysis and conceptual designs. Apl/jhu cp 051/tpr 035, Johns Hopkins University Applied Physics Laboratory, 1976.

[CSB77] H.Y. Chiu, G.B. Stupp, and S.J. Brown. Vehicle-follower control with variable-gains for short-headway automated guideway transit systems. *J. Dyn. Sys. Meas. Contr.*, 3(99):183–189, 1977.

[CT94] W. Chee and M. Tomizuka. Lane change maneuver for ahs applications. In *Proc. International Symposium on Advanced Vehicle Control*, pages 420–425, Japan, October 1994.

[DAP96] P. Daviet, S. Abdou, and M. Parent. Platooning for vehicles and automatic parking by scheduling robotic actions. In *International Symposium on Robotics and Manufacturing, WAC*, Montpellier, France, June 1996.

[Dar87] W. Darenberg. Automatische spurf´uhrung von kraftfahrzeugen. *Automobil-Industrie*, February 1987.

[Dav93] P. Daviet. Développement d'un capteur de position relative pour la formation de trains de véhicules. Rapport de DEA (M.I.S.I), Juin 1993.

[Dav98] P. Daviet. *Commande d'un Train de Véhicules Autonomes sans lien Matériel*. PhD thesis, Université de Dauphine, 1998. en préparation.

[DK90] M.E. Drummond and L.P. Kaelbling. Integrated agent architectures : benchmark tasks and evaluation metrics. Technical report, Teleos Research, Palo Alto, Calif., 1990.

[DP95] P. Daviet and M. Parent. Platooning for small public urban vehicles. In *International Symposium on Experimental Robotics*, Stanford, June 1995.

[DP96a] P. Daviet and M. Parent. Longitudinal and lateral servoing of vehicles in a platoon. In *Proc. Intelligent Vehicles '96 IEEE Symposium*, Japon, September 1996.

[DP96b] P. Daviet and M. Parent. Platooning technique for empty vehicles in the praxitele project. In *4th IEEE Mediterranean Symposium on New Directions in Control and Automation*, Chania, Crete, June 1996.

[DUC87] G. DUC. Théorie de la commande temporelle des systèmes linéaires. Ecole Supérieure d'Electricité, 1987.

[Dut92] B. Dutertre. *Spécification et Preuve de Systèmes Dynamiques : Application à* SIGNAL. PhD thesis, Université de Rennes, 1992.

[DZ86] E.D. Dickmanns and A. Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *Proc. SPIE*, volume 727, pages 161–168, 1986.

[DZ87] E.D. Dickmanns and A. Zapp. Autonomous high-speed road vehicle guidance by computer vision. In *Tenth IFAC World Congress*, volume 4, pages 232–237, Munich, 1987.

[EJAK95] B. Espiau, M. Jourdan, S. Abdou, and K. Kappellos. Specify, validate and implement a virtual "train" of vehicles. In *28th International Symposium on Automotive Technology and Automation*, (Stuttgart) Germany, September 1995.

[EW74] R.R. Evans and R.P. Whitten. Analysis and demonstration of position error headway protection for prt systems. In *Department of Audio Visual Extension, University of Minnesota, MN*, pages 335–348. Personal Rapid Transit II, 1974.

[FAS94] Y. Fujita, K. Akuzawa, and M. Sato. Radar brake system. In *Proc. International Symposium on Advanced Vehicle Control*, Tsukuba, Japan, October 1994.

[FB94a] P. Fancher and Z. Bareket. Evaluation headway control using range versus range-rate relationships. *Veh. Sys. Dyn.*, 8(23):575–596, 1994.

[FB94b] P. Fancher and Z. Bareket. The influence of intelligent cruise control systems on traffic flow. In *Proc. International Symposium on Advanced Vehicle Control*, Tsukuba, Japan, October 1994.

[FBJ94] P. Fancher, Z. Bareket, and G. Johnson. Predictive analyses of the performance of a headway control system for heavy commercial vehicles. In *Proc. 13th IAVSD Symposium*, pages 575–596, Chengdu, China, August 1994.

[FC77] R.E. Fenton and K.C. Chu. On vehicle automatic longitudinal control. *Trans. Sci*, 11(1):73–91, 1977.

[FEB+94] P.S. Fancher, R.D. Ervin, Z. Bareket, G.E. Johnson, M. Trefalt, J. Tiedecke, and W. Hagleitner. Intelligent cruise control : performances studies based upon an operating prototype. In *Proc. IVHS America 1994 Annual Meeting*, pages 391–399, Atlanta, Georgia, April 1994.

[Fen68] R.E. Fenton. One approach to highway automation. *IEEE Proc.*, 4(56):556–566, 1968.

[FGM+62] L.E. Floy, Gray, Morely, Pike, and Caulton. Electronic techniques in a system of highway vehicle control. *RCA Rev.*, 23(3):293–310, 1962.

[FGM+92] J.Cl. Fernandez, H. Garavel, L. Mounier, A. Rasseand, C. Rodriguez, and J. Sifakis. A tool box for the verification of lotos programms. In *International Conference On Software Engineering*, may 1992.

[Fir87] J. Firby. An investigation into reactive planning in complex domains. In *the AAAI 1987 Press*, pages 202–206, Menlo Park, Calif., 1987.

[Fir89] R.J. Firby. Adaptive execution in complex dynamic worlds. Technical Report YALEU/CSD RR 672, Yale University, 1989.

[Fir95] J. Firby. An architecture for a synthetic vacuum cleaner. In *AAAI Spring Symposium: Lessons Learned from Implemented Software Architectures for Physical Agents*. Stanford University, March 1995. available via anonymous ftp at hobbes.jsc.nasa.gov/pub/korten/spring_symposium/submissions.

[FM91] R.E. Fenton and R.J. Mayhan. Automated highway studies at the ohio state university an overview. *IEEE Trans. on Veh. Tech.*, 40(1):100–113, 1991.

[FMO76] R.E. Fenton, G.C. Melocik, and K.W. Olson. On the steering of automated vehicles : Theory and experiment. *IEEE Trans. Auto. Control*, 3(21), 1976.

[FO72] R.B. Fling and C.L. Olson. An integrated concept for propulsion, braking, control and switching of vehicles operating at close headways. In *Personal Rapid Transit*, pages 361–382. Institute of Technology, University of Minnesota, 1972.

[FOB71] R.E. Fenton, K.W. Olson, and J.G. Bender. Advances toward the automatic highway. In *Highway Research Record*, 344, pages 1–20, 1971.

[Fos79] E.R. Foster. A constant separation control law for use in automated guideways transit. In *Proc. 29th IEEE Vehicular Technology Conference*, pages 358–361, Arlington Heights, IL, March 1979.

[FS88] R.E. Fenton and I. Selim. On the optimal design of an automotive lateral controller. *IEEE Trans. Veh. Tech.*, 3(37):108–113, 1988.

[FS94] T. Fujioka and K. Suzuka. Lateral autonomous driving by sliding control. In *Proc. International Symposium on Advanced Vehicle Control*, pages 432–437, Japan, October 1994.

[FYT93] T. Fujioka, K. Yoshimoto, and S. Takaba. A case study on an automated driving highway system in japan. In *Transportation Research Board 72nd Annual Meeting*, Washington, D.C, Junary 1993.

[GAC87] M. Ghallab, R. Alami, and R. Chatila. Dealing with time in planning and execution monitoring. In *the Fourth International Symposium*, Santa Cruz (USA), August 1987.

[Gar60] K. Gardels. Automatic car controls for electronic highways. Technical Report GMR-276, General Motors Research Laboratories, 1960.

[Gar76] D.A. Gary. *Personal rapid transit III*. University of Minnesota, 1976.

[Gar77] W.L. Garrard. Longitudinal control for automated guideway transit vehicles. Technical report, University of Minnesota, July 1977. Report UMTA-MN-11-0002-77-2.

[Gar78] W.L. Garrard. Effects of jerk limiting on the stability of automated transit vehicles. *J. Dyn. Sys. Meas. Contr.*, 4(100):298–301, 1978.

[Gar95] P. Garnier. *Contrôle d'exécution réactif de mouvements de véhicules en environnement dynamique structuré*. PhD thesis, Institut National Polytechnique de Grenoble, décembre 1995.

[Gat91] E. Gat. Alfa : A language for programming reactive robotic control systems. In *the IEEE Conference on Robotics and Automation*, 1991.

[GC93] L. Giubbolini and C. Cugiani. radar systems for the control of the lateral position of the vehicle traveling along a motorway. In *Proc. International Symposium on Automotive Technology and Automation*, pages 689–696, Aachen, Germany, 1993.

[Ger93] M.W. Gertz. *A Multilevel Human-Machine Interface for Real-Time Sensor-Based Systems*. Ph. d. prospectus, Department of ECE, The Robotics Institute at Carnergie Mellon University, April 1993.

[GG89] D.M. Grimes and C.A. Grimes. Cradar - an open-loop extended-monopulse automotive radar. *IEEE Tras. Veh. Tech.*, 3(38):123–131, 1989.

[GGBM91] P. Le Guernic, T. Gautier, M. Le Borgne, and C. Le Maire. Programming real-time applications with SIGNAL. *Proceeding of the IEEE*, 79:1321–1336, September 1991.

[GK73a] W.L. Garrard and A. Kornhauser. Design of optimal feedback systems for longitudinal control of automated transit vehicles. *Trans. Res.*, 2(7):125–144, 1973.

[GK73b] W.L. Garrard and A. Kornhauser. Use of state observers in the optimal feedback control of automated transit vehicles. *J. Dyn. Sys. Meas. Contr.*, 2(95):220–227, 1973.

[GL87] M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *the Sixth National Conf. on Artificial Intelligence*, Seattle, Washington, July 1987.

[GL94a] M. Ghallab and H. Laruelle. Representation and control in ixtet, a temporal planner. In *the 2nd Int. Conf. on Artificial Intelligence Planning Systems*, June 1994.

[GL94b] D.N. Godbole and J. Lygeros. Longitudinal control of the lead car of a platoon. In *Proc. American Control Conference*, pages 398–402, Baltimore, MD, June 1994.

[Gon94] D. Gajski F. Vahid S. Narayan J. Gong. *Specification and Design of Embedded Systems*. Ed Prentice-Hall, 1994.

[Gra93] V. Graefe. Vision for intelligent road vehicles. In *Proc. Intelligent Vehicles '93 IEEE Symposium*, pages 135–140, Tokyo, July 1993.

[GUA94] J. Guldner, V.I. Utkin, and J. Ackermann. A sliding mode control approach to automatic car steering. In *Proc. American Control Conference*, pages 1969–1973, Baltimore, MD, June 1994.

[Hal93] N. Halbwachs. *Sybchronous Programming of Reactive Systems*. Kluwer Academic, 1993.

[Han66] M.E. Hanson. *Project METRAN : An Integrated, Evolutionary Transportation System for Urban Areas*. MIT Press, 1966.

[Har87] D. Harel. Statecharts : A visual approch to complex system. *Science of Computer Programming*, 8:231–275, 1987.

[Hes72] R. Hesse. Cabinentaxi - a personal public transportation system. In *Trans. 1,3*, pages 321–329, 1972.

[Hin75] E.J. Hinman. Command and control studies for personal rapid transit, program status 1975. Technical Report 178, Johns Hopkins University, Applied Physics Laboratory, April 1975. APL/JHU CP 039/TPR 030.

[HLR92] N. Halbwachs, F. Lagnier, and Ch. Ratel. Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE. *IEEE Transactions on Software Engineering*, pages 785–793, September 1992.

[HLR93] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *Third Int. Conf. on Algebraic Methodology and Software Technology, AMAST'93*. Twente Workshops in Computing, Springer Verlag, juin 1993.

[HNS91] J.K. Hedrick, D.H. McMahoncand V. Narendran, and D. Swaroop. Longitudinal vehicle controller design for ivhs systems. In *Proc. 1991 American Control Conference*, pages 3107–3112, Boston, MA, June 1991.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. logic and models of concurrent systems. In Springer Verlag, editor, *NATO Advanced Study Institute on Logics and Models for Verification and Specification of Concurrent Systems*, volume 13, 1985.

[HR76] L.L. Hoberock and R.J.Jr. Rouse. Emergency control of vehicle platoons : system operation and platoon leader control. *J. Dyn. Sys. Meas. Contr.*, 3(98):245–251, 1976.

[HR92] T. Heikkila and J. Roning. Pem modelling : A framework for designing intelligent robot control. *Journal of Robotics and Mechatronics*, 4(5):432–444, 1992.

[HS94] J.K. Hedrick and D. Swaroop. Dynamic coupling in vehicles under automatic control. In *Proc. 13th IAVSD Symposium*, pages 209–220, Chengdu, China, August 1994.

[HSEV91] A. Hsu, S. Sachs, F. Eskafi, and P. Varaiya. The design of platoon maneuvers for ivhs. In *Proc. 1991 American Control Conference*, pages 2545–2550, Boston, MA, June 1991.

[HT94] T. Hessburg and M. Tomizuka. Fuzzy logic control for lateral vehicle guidance. *IEEE Contr. Sys. Mag.*, 4(14), 1994.

[HY90] K. Hayafune and H. Yoshida. Control method of autonomous vehicle considering compatibility of riding comfort and vehicle controllability. In *Automated Highway/Intelligent Vehicle Systems : Technology and Socioeconomic Aspects*, pages 35–40, August 1990.

[ICAR95] Ingrand, R. Chatila, R. Alami, and F. Robert. Embedded control of autonomous robot using procedural reasoning. In *Proceeding of the International Conference on Robotics and Automation*, Nagoya, Japan, 1995.

[Ins68] Transportation Research Institute. Urban rapid transit concepts and evaluation. TRI Research Report 1, Carnegie-Mellon University, Pittsburgh, 1968.

[Int78] International Conference on Advanced Transportation and Urban Revitalization. *Advanced Transit Association*, Indianapolis, IN, May 1978.

[IS85] J. Ish-Shalom. The cs language concept : A new approach to robot motion design. *Int. Jouranl of Robotics Res.*, 1(4):42–58, 1985.

[Ish74] T. Ishii. The control system of cvs using the two-target tracking scheme. In *Personal Rapid Transit II, Dept. of Audio Visual Extension*, pages 325–334, University of Minnesota, 1974.

[ISK89] J. Ish-Shalom and P. Kazanzides. Sparta : Multiple signal processors for high-performance robot control. In *IEEE Transactions on Robotics and Automation*, volume 5, october 1989.

[IX94] P.A. Ioannou and Z. Xu. Throttle and brake control systems for automatic vehicle following. *IVHS J.*, 1(4):345–377, 1994.

[JAL79] A.R. Johnston, T. Asseli, and J.Y. Lai. Automated vehicle guidance using discrete reference markers. *IEEE Trans. Veh. Tech.*, 1(28):95–106, 1979.

[JLMR94] M. Jourdan, F. Lagnier, F. Maraninchi, and P. Raymond. A multiparadigm language for rearive systems. In *Proceeding of the IEEE Int. Conf. on Computer Languages, ICCL'94*, Toulouse, France, Mai 1994.

[Joh93] B. Johnson. A lateral guidance radar module. In *Proc. International Symposium on Automotive Technology and Automation*, pages 697–703, Aachen, Germany, 1993.

[Jou94] M. Jourdan. *Etude d'un environnement de programmation et de vérification des systèmes réactifs, multi-langages et multi-outils*. PhD thesis, Université Joseph Fourier de Grenoble I, 1994.

[Jou95] M. Jourdan. Integrating formal verification methods of quantitative real time properties into a development environment for robot controllers. Technical report 2540, INRIA, 1995.

[KAJE95] K. Kappellos, S. Abdou, M. Jourdan, and B. Espiau. Specification, formal verification and implementation of tasks and mission for an autonomous vehicle. In *International Symposium on Experimental Robotics*, Stanford (USA), June 1995.

[Kap94] K. Kapellos. *Environnement de Programmation des Applications Robotiques Réactives*. PhD thesis, Ecole des Mines de Paris, 1994.

[KIN94] H. Kikuchi, M. Ishiyama, and T. Nakajima. Development of laser radar for radar brake system. In *Proc. International Symposium on Advanced Vehicle Control*, pages 385–389, Japan, October 1994.

[KKT84] T. Kanade, P.K. Khosla, and N. Tanaka. Real-time control of the cmu direct drive arm ii using customized inverse dynamics. In *Proc. of the 23rd IEEE Conference on Decision and Control*, pages 1345–1352, Las Vegas, NV, December 1984.

[KLG90] N. Kehtarnavaz, J.S. Lee, and N.C. Griswold. Vision-based convoy following by recursive filtering. In *Proc. 1990 American Control Conference*, pages 268–273, San Diego, May 1990.

[Kor74] A. Kornhauser. Optimal sample-data control of prt vehicles. In *Department of Audio Visual Extension, University of Minnesota, MN*, pages 359–366. Personal Rapid Transit II, 1974.

[KR89] L.P. Kaelbling and S. Rosenschein. Action and planning in embedded agents. Teleos Research TR-89-07, Menlo Park, Calif., 1989.

[KVW90] U. Karaaslan, P. Varaiya, and J. Walrand. Two proposals to improve freeway traffic flow. In *Proc. 1991 American Control Conference*, pages 2539–2544, Boston, MA, June 1990.

[LA68] A. Levis and M. Athans. Sampler-data control of high-speed trains. Technical Report ESL-R-339, PB 177 669, M.I.T Electronic Systems Laboratory, January 1968.

[LB94] W.A. Leasure and A.L. Burgett. Nhtsa's ivhs collision avoidance research program : strategic plan and status update. In *Proc. First World Congress on Applications of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2216–2223, Paris, November-December 1994.

[Ler98] W. Leroquais. *Modélisation et commande de robots mobiles à roues en présence de glissements.* PhD thesis, Ecole des Mines de Paris, 1998. en préparation.

[LG94] J. Lygeros and D.N. Godbole. An interface between continuous and discrete-event controllers for vehicle automation. In *Proc. 1994 American Control Conference*, pages 801–805, Baltimore, MD, June 1994.

[Mar90] F. Maraninchi. ARGOS *: un langage graphique pour la conception et la validation des systèmes réactifs*. PhD thesis, Université Joseph Fourier, Grenoble, 1990.

[Mas92] V. Masaki. *Vision-Based Vehicle Guidance*. Springer Verlag, New York, 1992.

[Mau74] J.P. Maury. *The ARAMIS PRT system*. SAE Paper 740143, 1974.

[MB82] R.J. Mayhan and R.A. Bishell. A two-frequency radar for vehicle automatic lateral control. *IEEE Trans. Veh. Tech.*, 1(31):32–39, 1982.

[Met88] H.G. Metzler. Computer vision applied to vehicle operation. In *SAE Technical Paper Series, SAE, Warrendale. PA. Futur Transportation Technology Conference and Exposition*, 881167, San Fransisco, August 1988.

[Mey88] A. Meystel. Intelligent control in robotics. *Journal of Robotic Systems*, 1988.

[MGH94] D.B. Maciuca, J.C. Gerdes, and J.K. Hedrick. Automatic braking control for ivhs. In *Proc. International Symposium on Advanced Vehicle Control*, pages 396–401, Tsukuba, Japan, October 1994.

[MHS90] D.H. McMahon, J.K. Hedrick, and S.E. Shladover. Vehicle modeling and control for automated highway systems. In *Proc. 1990 American Control Conference*, pages 297–303, San Diego, May 1990.

[Miq85] R. Miquel. *Le filtrage numerique par microprocesseurs*. Editests, Lagny-sur-Marne. PSI, 1985.

[ML91] D. Miller and R.C. Lennox. An object-oriented environment for robot system architectures. In *in Proc. of 1990 IEEE International Conference on Robotics and Automation*, pages 352–864, Cincini, Ohio, May 1991.

[MMP92] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In Springer-Verlag editor, editor, *Real Time : Theory in Practice*, pages 447–484, 1992.

[MN92] R. Muller and G. Nocker. Intelligent cruise control with fuzzy logic. In *Proc. Intelligent Vehicles 92 Symposium, IEEE*, pages 173–178, 1992.

[Mou92] L. Mounier. *Méthodes de Vérification de Spécifications Comportementales :étude et mise en oeuvre*. Thèse de doctorat, Université Joseph Fourier, Grenoble, 1992.

[Mou95] C. Mounet. Câblage des ultrasons. Rapport de stage (E.P.F), Juillet-Août 1995.

[NH94] V.K. Narendran and J.K. Hedrick. Autonomous lateral control of vehicles in an automated highway system. *Veh. Sys. Dyn.*, 4(23):307–324, 1994.

[NKYA94] M. Nakamura, H. Kawashima, T. Yoshikai, and K. Aoki. Road-vehicle cooperation driving system. In *Proc. 1994 Vehicle Navigation and Information Systems*, pages 425–430, Yokohama, Japan, August-Setember 1994.

[NR94] K. Naab and G. Reichart. Driver assistance systems for lateral and longitudinal vehicle guidance - heading control and active cruise support. In *Proc. International Symposium on Advanced Vehicle Control*, pages 449–454, Tsukuba, Japan, October 1994.

[NS90a] A. Niehaus and R.F. Stengel. An expert system for automated highway driving. In *Proc. 1990 American Control Conference*, pages 274–280, San Diego, CA, May 1990.

[NS90b] A. Niehaus and R.F. Stengel. Rule-based guidance for vehicle highway driving in the presence of uncertainty. In *Proc. 1991 American Control Conference*, pages 3119–3124, Boston, MA, June 1990.

[NS91] A. Niehaus and R.F. Stengel. Probability-based decision making for automated highway driving. In *Proc. Vehicle Navigation and Information Systems Conference*, pages 1125–1136, Dearborn, MI, October 1991.

[NS97] R. Nikoukhah and S. Steer. Scicos a dynamic system builder and simulator user's guide - version 1.0. Technical report RT-0207, INRIA, juin 1997.

[NSH89] S. Narasimhan, D. Siegel, and J. Hollerbach. Condor : An architecture for controlling the utah-mit dexterous hand. In *in the IEEE Transactions on Robotics and Automation*, 1989.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into timed automata. In *IEEE Transactions on Software Engineering, special issue on Specification and Analysis of Real-Time Systems*, 1992.

[Occ84] *The* OCCAM *programming manual*, prenrtice hall edition, 1984.

[OG79] D.E. Olson and W.L. Garrard. Model-follower longitudinal control for automated guideway transit vehicles. *IEEE Trans. Veh. Tech.*, 1(28):36–45, 1979.

[OKF90] A. Okuno, A. Kutami, and K. Fujita. Towards autonomous cruising on highways. In *Automated Highway/Intelligent Vehicle Systems : Technology and Socioeconomic Aspects*, pages 7–16. SAE Publication SP-833, August 1990.

[OKI92] K. Ohnishi, J. Komura, and T. Ishibashi. Development of automatic driving system on rough road - realisation of high reliable automatic driving system. In *Proc. Intelligent Vehicles 92 Symposium, IEEE*, Detroit, MI, June-July 1992.

[Ols69] K.W. Olson. A system for automatic vehicle lateral guidance. In *Highway Research Record*, 275, pages 1–11, 1969.

[Ols77] K.W. Olson. Wire-reference configurations in vehicle lateral control. *IEEE Trans. Veh. Tech.*, 2(26):161–172, 1977.

[Osh65] R. Oshima. Control system for automobile driving. In *Proc. Tokyo IFAC Symposium*, pages 347–357, 1965.

[OTO92] N. Ooka, T. Tsuboi, and H. Oka. Development of automatic driving system on rough road - fault tolerant structure for electronic controller. In *Proc. Intelligent Vehicles 92 Symposium, IEEE*, Detroit, MI, June-July 1992.

[P.93] Varaiya P. Smart cars on smart roads : Problems of control. *IEEE Trans. on Auto. Contr*, 2(38):195–207, 1993.

[Pal93] U. Palmquist. Intelligent cruise control and roadside information. *IEEE Micro*, 1(13):20–28, 1993.

[PAQ69] E.T. Powner, J. Anderson, and G. Qualtrough. Optimal digital control of cascaded vehicles in high-speed transportation systems. *Trans. Res.*, 1(3), 1969.

[Par92] J.P. Paris. *Exécution de tâches asynchrones depuis* ESTEREL. Thèse de doctorat, Université de Nice, 1992.

[Pas71] S. Pasternack. Modern control aspects of automatically steered vehicles. Technical note, DOT-TSC-OST, December 1971.

[Pas73] S. Pasternack. Some modern control aspects of automatically steered rubber tired vehicles. In *Second Intersociety Conference on Transportation*, September 1973.

[Pay86] D. Payton. An architecture for reflexive autonomous vehicle control. In *in Proc. IEEE International Conference on Robotics and Automation*, pages 1838–1845, San Fransisco, CA, April 1986.

[PDA95] M. Parent, P. Daviet, and S. Abdou. Vision technique for platoon driving. In *Application of Advanced Technologies in Transportation Conference*, Capri, Italy, June 1995.

[PDDM94] M. Parent, P. Daviet, J.C. Denis, and T. M'Saada. Automatic driving in stop and go traffic. In *Conference IEEE Intelligent Vehicles'94*, Paris, October 1994.

[Pen] S. Penoyre. Dual mode road vehicles. *Traff. Eng. Contr.*, 13(5):189–191.

[Pen93] H. Peng. Highway-level vehicle control for ahs. *IVHS J.*, 1(2):181–190, 1993.

[PG72] L.E. Peppard and V. Gourishankar. An optimal automatic car-following system. *IEEE Trans. Veh. Tech.*, 2(21):67–73, 1972.

[PH73] G.L. Pitts and E. Hinman. Augmented block guidance a control concept for automated urban transit. *APL Tech. Diag.*, 12(1):2–10, 173.

[PHT94] H. Pham, K. Hedrick, and M. Tomizuka. Autonomous steering and cruise control of automobiles via sliding mode control. In *Proc. International Symposium on Advanced Vehicle Control*, pages 444–448, Tsukuba, Japan, October 1994.

[Pit71] G.L. Pitts. A collision avoidance control law for fixed block, short-headway transportation systems. In *Control and Flight Mechanics Conference*, pages 71–942, Hempstead, NY, August 1971.

[Pit72] G.L. Pitts. Augmented block guidance for short-haul transpostation systems. Technical report, Johns Hopkins University Applied Physics Laboratory, September 1972. APL JHU CP 019/TPR 023.

[PL96] I. E. Paromtchik and C. Laugier. Autonomous parallel parking of an nonholonomic vehicle. In *Proceeding of the IEEE/IES Symposium Vehicles'96*, Tokyo, Japan, September 1996.

[PRK90] T. Pilutti, U. Raschke, and Y. Koren. Computerized defensive driving rules for highway maneuvers. In *Proc. 1990 American Control Conference*, pages 809–811, San Diego, CA, May 1990.

[Pro58] Proc. Highway Research Board 37th Annual Meeting. *Electronic control of motor vehicles on the higways*, 1958.

[PT90] H. Peng and M. Tomizuka. Vehicle lateral control for highway automation. In *Proc. 1990 American Control Conference*, pages 788–794, San Diego, May 1990.

[PT91] H. Peng and M. Tomizuka. Preview control for vehicle lateral guidance in highway automation. In *Proc. 1991 American Control Conference*, pages 3090–3095, Boston, MA, June 1991.

[PT93] M. Parent and P.Y. Texier. Le transport public individuel. In *Colloque SATCAR*, Clermont-Fd, Mai 1993.

[PTLR94] D. Pomerleau, C. Thorpe, D. Longer, and J.K. Rosenblatt. Avcs research at carnegie mellon university. In *Proc. IVHS America 1994 Annual Meeting*, pages 257–261, Atlanta, Georgia, April 1994.

[PTZD94] S. Patwardhan, M. Tomizuka, W.B. Zhang, and P. Devlin. Theory and experiments of tire blow-out effects and hazard reduction control for automated vehicle lateral control system. In *Proc. International Symposium on Advanced Vehicle Control*, pages 426–431, Japan, October 1994.

[Pue77a] A.J. Pue. A state-constrained approach to vehicle-follower control for short-headway agt systems. Apl/jhu cp 058/tpr 038, Johns Hopkins University Applied Physics Laboratory, August 1977.

[Pue77b] A.J. Pue. A state-constrained approach to vehicle-follower control for short-headway automated transit systems. In *Proc. 1977 Joint Automatic Control Conference*, volume 1, pages 401–407, San Fransisco, June 1977.

[Pue78] A.J. Pue. A state-constrained approach to vehicle-follower control for short-headway automated transit systems. *J. Dyn. Sys. Meas. Contr.*, 4(100):291–297, 1978.

[Pue79] A.J. Pue. Implementation trade-offs for a short-headway vehicle - follower automated transit system. *IEEE Trans. Veh. Tech.*, 1(28):46–55, 1979.

[PZS+93] H. Peng, W.B. Zhang, S.E. Shladover, M. Tomizuka, and A. Arai. Magnetic-marker-based lane keeping : a robustness experimental study. In *SAE Congress*, pages 127–132, Detroit, MI, February 1993.

[PZS94] H. Peng, W.B. Zhang, and S.E. Shladover. A reusability study of vehicle lateral control system. *Veh. Sys. Dyn.*, 4(23), 1994.

[RCCE92] O. Roux, D. Creusot, F. Cassez, and J.P. Elloy. Le langage réactif asynchrone ELECTRE. *Technique et Science Informatiques (TSI)*, 11(5):35–66, 1992.

[RdS90] V. Roy and R. de Simone. Auto and autograph. In *In R. Kurshan, editor, proceedings of Workshop on Computer Aided Verification*, New Brunswick, June 1990.

[Rea93] Ready Systems, Inc. *VRTX User's Manual*, 1993.

[Res72] Road Research. *Annual Report of the Road Research Laboratory*. Her Majesty's Stationery Office, 1972.

[RG94] W. Ren and D. Green. Continuous platooning : a new evolutionary operating concept for automated highway systems. In *Proc. American Control Conference*, pages 21–25, Baltimore, MD, June 1994.

[RH76] R.J.Jr. Rouse and L.L. Hoberock. Emergency control of vehicle platoons : control of following-law vehicles. *J. Dyn. Sys. Meas. Contr.*, 3(98):239–244, 1976.

[RHM73] A. Rahimi, L.P. Hajdu, and H.L. Macomber. Safety analysis for automated transportation systems. *Trans. Sci*, 7(1):1–17, 1973.

[Ric66] B. Richards. New movement in cities. *Studio Vista*, page 96, 1966.

[RMPB94] D. Ramaswamy, J.V. Medanic, W.R. Perkins, and R. Benekohal. Lane assignment on an automated highway. In *Proc. 1994 American Control Conference*, pages 413–417, Baltimore, MD, June 1994.

[RN94] G. Reichart and K. Naab. Systems for lateral and longitudinal vehicle guidance. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2126–2133, Paris, November-December 1994.

[RRSV87] J.L. Richier, C. Rodriguez, J. Sifakis, and J. Voiron. XESAR *: A Tool for Protocol Validation.User's Guide*. LGI-IMAG, Grenoble, France, 1987.

[RS90] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[RTJV94] J. Roning, T. Taipale, J.Okkonen, and P. Vaha. The machine of the future. Technical report, Federation of Flnnish Metal, Engineering and Electrotechnical Industries, FIMET, Helsinki, Finland, 1994.

[RV94a] B.S.Y. Rao and P. Varaiya. Potential benefits of roadside intelligence for flow control in an ivhs. In *Proc. 1994 American Control Conference*, pages 418–422, Baltimore, MD, June 1994.

[RV94b] B.S.Y. Rao and P. Varaiya. Roadside intelligence for flow control in an intelligent vehicle and highway system. *Trans. Res. C : Emerg. Tech.*, 2C(1):49–72, 1994.

[RVE93a] B.S.Y. Rao, P. Varaiya, and F. Eskafi. Flow benefits of autonomous intelligent cruise control in mixed manual and automated traffic. *Trans. Res. Rec 1408*, pages 36–43, 1993.

[RVE93b] B.S.Y. Rao, P. Varaiya, and F. Eskafi. Investigations into achievable capacities and stream stability with coordinated intelligent vehicles. *Trans. Res. Rec 1408*, pages 27–35, 1993.

[Sam93] C. Samson. Commande de véhicules non-holonomes pour le suivi de trajectoire et stabilisation à une position désirée. In *Colloque Automatique pour les véhicules terrestres*, Amiens, 1993.

[Sax93] L. Saxton. Mobility 2000 and the roots of ivhs. In IVHS Rev., editor, *Spring*, pages 11–26, 1993.

[SB94] L. Saxton and J.R. Bishop. The united states dot automated higway system (ahs) program. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 1952–1958, Paris, November-Dcember 1994.

[SBE91] C. Samson, M. Le Bourgne, and B. Espiau. *Robot Control : the Task-Function Approach*. Clarendon Press, Oxford Science Publications, U.K., 1991.

[SBP93] S.E. Shladover, R. Bushey, and R.E. Parsons. California and the roots of ivhs. In IVHS Rev., editor, *Spring*, pages 27–34, 1993.

[SBS79] S.J. Sklar, J.P. Bevans, and G. Stein. "safe-approach" vehicle-follower control. *IEEE Trans. Veh. Tech.*, 1(28):56–62, 1979.

[Sch87] M.J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *In Proceeding of the Tenth International Joint Conference on Artificial Intelligence (IJCAI'87)*, Milan, Italy, August 1987.

[Sch89] M.J. Schoppers. In defense of reaction - plans as caches. *Al Magazine, Winter*, pages 51–60, 1989.

[SCPC95] S.A. Schneider, V. W. Chen, and G. Pardo-Castellote. The controlshell component-based real-time programming system. In *IEEE International Conference on Robotics and Automation*, 1995.

[SD90] S. Sheikholeslam and C.A. Desoer. Longitudinal control of a platoon of vehicles. In *Proc. 1990 American Control Conference*, pages 291–296, San Diego, May 1990.

[SD91] S. Sheikholeslam and C.A. Desoer. Longitudinal control of a platoon of vehicles with no communication of lead vehicle information. In *Proc. 1991 American Control Conference*, pages 3102–3106, Boston, MA, June 1991.

[SD92a] S. Sheikholeslam and C.A. Desoer. Combined longitudinal and lateral control of a platoon of vehicles. In *Proc. 1992 American Control Conference*, pages 1763–1767, Chicago, IL, June 1992.

[SD92b] S. Sheikholeslam and C.A. Desoer. A system level study of the longitudinal control of a platoon of vehicles. *J. Dyn. Sys. Meas. Contr.*, 2(114):286–292, 1992.

[SECK93] D. Simon, B. Espiau, E. Castillo, and K. Kapellos. Computer-aided design of a geniric robot controller handling reactivity and real-time control issues. *IEEE Trans. on Control Systems Technology*, 1(4), December 1993.

[SEKPG97] D. Simon, B. Espiau, K. Kapellos, and R. Pissard-Gibollet. Orccad: Software engineering for real-time robotics, a technical insight. *Robotica*, 15(2), 1997.

[SFC94] D. Simon, P. Freedman, and E. Castillo. Analysing the temporal behavior of real time closed-loop robotic tasks. In *IEEE International Conference on Robotics and Automation*, San Diego, May 1994.

[SFWR77] S.E. Shladover, R. Fish, D.N. Wormley, and H.H. Richardson. Performance of steering controllers for automated guideway transit vehicles. In *2nd International Union of Theorical and Applied Mechanics Symposium Dynamics of Vehicles on Roads and Tracks. Swets and Zeitlinger*, pages 127–135, Amsterdam, 1977.

[SH79] R.R. Smisek and G.A. Harder. Automated control of guideway transit vehicles. In *29th IEEE Vehicular Technology Conference, Arlington Heights*, Mars 1979.

[SHCI94] D. Swaroop, J.K. Hedrick, C.C. Chien, and P. Ioannou. A comparison of spacing and headway control laws for automatically controlled vehicles. *Veh. Sys. Dyn.*, 8(23):597–625, 1994.

[Shl76] S.E. Shladover. Performance limits of automatic steering controllers. In *Institut of Technology, University of Minnesota, MN*, pages 233–245. Personal Rapid Transit III, 1976.

[Shl78] S.E. Shladover. Longitudinal control of automated guideway transit vehicles within platoons. *J. Dyn. Sys. Meas. Contr.*, 4(100):302–310, 1978.

[Shl91] S.E. Shladover. Longitudinal control of automative vehicles in close-formation platoons. *J. Dyn. Sys. Meas. Contr.*, (113):231–241, 1991.

[Shl92] S.E. Shladover. The california path program of ivhs research and its approach to vehicle-highway automation. In *Proc. Intelligent Vehicle 92 Symposium*, pages 347–352, Detroit, MI, June-July 1992.

[SHR72] D.E. Stepner, L.P. Hajdu, and A. Rahimi. Safety considerations for personal rapid transit. Technical Report 457-472, Personal Rapid Transit, Institute of Technology, University of Minnesota, 1972.

[SHW92] T. Shigematu, Y. Hashimoto, and T. Watanabe. Development of automatic driving system on rough road - automatic steering control by fuzzy algorithm. In *Proc. Intelligent Vehicles 92 Symposium, IEEE*, Detroit, MI, June-July 1992.

[Sim90] R. Simmons. Concurrent planning and execution for the walking robot. Technical report, Robotics Institute, Carnegie Mellon University, 1990.

[Sim95] R. Simmons. Towards reliable autonmous agents. In *In Proceeding of the 1995 AAAI Spring Symposium : Lessons Learned from Implemented Software Architectures for Physical Agents*, Stanford University, March 1995. available via anonymous ftp at hobbes.jsc.nasa.gov/pub/korten/spring_symposium/submissions.

[SKHK89] D. Schmitz, P. Khosla, R. Hoffman, and T. Kanade. Chimera : a real-time programming environement. In *IEEE*, pages 846–852, 1989.

[SKR93] A. Saffioti, K. Konolige, and E. Ruspini. A multivalued logic approach to integrating planning and controle. Technical Note 533, SRI International, 333 Ravenswood Avenue, 1993. also to appear in Artificial Intelligence.

[Smi] P.G. Smith. Discrete-time longitudinal control of dual mode automobiles. *ASME Paper 73-ICT-77*, page 12.

[Sof89] Software Components Group, Inc., 4655 Old Ironsides Drive, Santa Clara, CA 95054. *pSOS+/68K Real-Time, Multi-processing Operating System Kernel User's Manual*, Juanary 1989.

[Sor96] Y. Sorel. Real time embedded image processing applications using the a3 methodology. In *IEEE International Conference on Image Processing*, Nice, France, November 1996.

[SP94] G. Sala and F. Pressi. Stategies for intelligent cruise control (icc) application. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2039–2046, Paris, November-December 1994.

[SPB87] K. Schwan, P.Gopinath, and W. Bp. Chaos : Kernel support for object in the real-time domain. In *IEEE Transaction on Computers*, volume C-36, pages 904–916, August 1987.

[Spr83] Springer Verlag. *The Programming Language* ADA *reference Manual*, 1983.

[SS62] R.A. Spangler and F.M. Snell. A new vehicle guidance and speed control system. In *Highway Research Record*, 338, pages 20–27, 1962.

[SS94] D.E. Smith and J.M. Starkey. Effects of model complexity on the performance of automated vehicle steering controllers : controller developement and evaluation. In *Veh. Sys. Dyn.*, number 8 in 23, pages 627–645, 1994.

[SSK89] D.B. Stewart, D.E. Schmitz, and P. Khosla. Chimera ii : a real-time multiprocessing environment for sensor-based robot control. In *the IEEE International Symposium on Intelligent Control*, Albany, NY, September 1989.

[ST94a] H. Satoh and I. Taniguchi. A new control method of adaptive cruise control. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2094–2101, Paris, November-December 1994.

[ST94b] T. Suzuki and A. Tachibana. A fully automated intelligent vehicle system based on computer vision for autmated highway system. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 1925–1932, Paris, November-December 1994.

[STAS94] H. Sekine, K. Tamura, N. Asanuma, and M. Sato. A study of electronic cornering speed control system. In *Proc. International Symposium on Advanced Vehicle Control*, pages 455–460, Tsukuba, Japan, October 1994.

[SVK93] D.B Stewart, R.A Volpe, and P.K Khosla. Design of dynamically reconfigurable real-time software using port-based objects. Technical Report CMU-RI-TR-93-11, CMU, 1993.

[SWRF78] S.E. Shladover, D.N. Wormley, H.H. Richardson, and R. Fish. Steering controller design for automated guideway transit vehicles. *J. Dyn. Sys. Meas. Contr.*, 1(100):1–8, 1978.

[Tes61] Driverless Bus Tested. Automotive industries. 9(125):29, 1961.

[THJ+93] Morita T., Takahashi H., Satonobu J., Kojima K., and Morimoto Y. An approach to the intelligent vehicle. In *Proc. Intelligent Vehicles 93 Symposium*, pages 426–431, Tokyo, Japan, July 1993.

[TM90] S. Tsugawa and S. Murata. Velocity control for vehicle following through vehicle/vehicle communication. In *Proc. 22nd International Symposium on Automotive Technology and Automation*, pages 343–350, Florence, Italy, May 1990.

[trw69] TRW systems group : Automated highway systems, December 1969. PB 191 696.

[Tsa94] H.S.J. Tsao. A functional architecture for automated highway traffic planning. In *Proc. IVHS America 1994 Annual Meeting*, volume 2, pages 597–605, Atlanta, Georgia, April 1994.

[Tsu93] S. Tsugawa. Vision-based vehicles in japan : the machine vision systems and driving control systems. In *Proc. IEEE International Symposium on Industrial Electronics*, Budapest, Hungary, June 1993.

[TWF91] S. Tsugawa, N. Watanabeand, and H. Fujii. Super smart vehicle system - its concept and preliminary works. In *Proc. Vehicle Navigation and Information Systems Symposium*, pages 269–277, Dearborn, October 1991. SAE Publication P-253.

[TYHM79] S. Tsugawa, T. Yatabe, T. Hirose, and S. Matsumoto. An automobile with artificial intelligence. In *Proc. Sixth International Joint Conference on Artificial Intelligence*, volume 2, Tokyo, Japan, August 1979.

[Ulm94] B. Ulmer. Autonomous automated driving in real traffic. In *Proc. First World Congress on Application of Transport Telematics and Intelligent Vehicle-Highway Systems*, pages 2118–2125, Paris, November-December 1994.

[VZ96] D. Van-Zwynsvoorde. Mise en oeuvre de capteurs ultrasonores pour la conduite autonome d'un vehicule electrique. Rapport du Projet de fin d'études D.E.A Automatique-Informatique Industrielle (INSA de Toulouse), Juin 1996.

[Wei88] W. Weidemann. Electronic lateral control of vehicles. In *SAE Technical Paper Series Warrendale. Futur Transportation Technology Conference and Exposition*, number 881166, San Fransisco, August 1988.

[Whi79] R.P. Whitten. Short station ramps for agt systems. *IEEE Tras. Veh. Tech.*, 1(28):63–70, 1979.

[Wil70] D. Wilkie. Moving cell control scheme for automated transportation systems. *IEEE Trans. on Veh*, 4:347–364, 1970.

[Win93] Wind River Systems, Inc, 1351 Ocean Ave. Emeryville, CA 94608. *VxWorks User's Manual*, 1988-1993.

[WM95] D.E. Wilkins and K.L. Myers. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 1995. (http://www.ai.sri.com/people/wilkens/papers.html).

[WMLW94] D.E. Wilkins, K.L. Myers, J.D. Lowrance, and L.P. Wesley. Planning and reaction in uncertain and dynamic environment. *Journal of Experimental and Theoretical AI*, 6:197–227, 1994. (http://www.ai.sri.com/people/wilkens/papers.html).

[WT72] D.E. Whitney and M. Tomizuka. Normal and emergency control of a string of vehicles by fixed reference sampled data control. Technical Report 383-404, Personal Rapid Transit, Institute of Technology, University of Minnesota, 1972.

[WW74] R.P. Whitten and R.D. Wright. Analysis of short ramps for dual-mode and prt stations. In *First International Conference on Dual-Mode Transportation*, Washington, D.C., May 1974.

[YG74] S.C. Yang and W.L. Garrard. A low sensitivity modern approach to the longitudinal control of automated transit vehicles. *J. Dyn. Sys. Meas. Contr.*, 2(96):218–228, 1974.

[YHTM89] T. Yatabe, T. Hirose, S. Tsugawa, and S. Murata. Control of autonomous vehicles with vehicle-to-vehicle communication. In *Proc. CCCT 89, IFAC Symposium*, pages 553–558, Paris, 1989.

[YM92] Y. Yasui and D.L. Margolis. Lateral control of automobiles using a look-ahead sensor. In *Proc. International Symposium on Advanced Vehicle Control*, pages 292–297, Yokohama, Japan, September 1992.

[Yov93] S. Yovine. *Méthode et Outils pour la Vérification Symbolique de Systèmes Temporisés*. Thèse de doctorat, Institut National Polytechnique de Grenoble, mai 1993.

[YTSAH92] K. Youcef-Toumi, Y. Sasage, J. Ardini, and S.Y. Huang. The application of time delay control to an intelligent cruise control system. In *Proc. 1992 American Control Conference*, Chicago, IL, June 1992.

[ZB93] X. Zhang and T. Benz. Simulation and evaluation of "intelligent cruise control". *IVHS J.*, 1(2):181–190, 1993.

[Zha91] X. Zhang. Intelligent driving - prometheus approaches to longitudinal traffic flow control. In *Proc. Vehicle Navigation and Information Systems Conference*, pages 999–1010, Dearborn, MI, October 1991.

[ZP90] W.B. Zhang and R.E. Parsons. An intelligent roadway reference system for vehicle lateral guidance/control. In *Proc. 1990 American Control Conference*, pages 281–286, San Diego, May 1990.