TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

RÉSUMÉ

Un système de navigation autonome ou robot mobile autonome doit être capable de percevoir son environnement, générer et exécuter une trajectoire en réaction appropriée à l'information perçue. Il faut pour cela le doter d'un système de navigation robuste.

Ce mémoire présente une étude sur le contrôle du cap et de la vitesse d'un véhicule au moyen d'un perceptron multicouche entrainé avec l'algorithme de rétro-propagation du gradient.

L'objectif est d'évaluer la capacité de ce type de réseau de neurones à contrôler le cap et la vitesse que doit adopter un robot mobile en fonction de ses entrées perceptives pour éviter d'entrer en collision avec les obstacles présents dans son voisinage.

Pour ce faire, une stratégie de perception et de planification du déplacement a été développée et sa capacité à contrôler un mobile autonome a été montrée. Cette stratégie utilise deux réseaux de neurones fonctionnant en parallèle, l'un pour la commande du cap et l'autre pour la commande de la vitesse exploitant les deux le même système perceptif.

Les résultats de la simulation sur Matlab sont présentés et analysés au regard de ce qu'est l'apprentissage statistique et de ce qu'on peut en attendre dans le cadre de la navigation autonome.

Cette approche a l'avantage d'être simple et d'indiquer avec précision les degrés de rotation et la vitesse nécessaire avec une vision à long terme empêchant les blocages ou divergences de parcours contrairement aux approches habituelles rencontrées dans la littérature.

Mots clés: télé-opération, navigation autonome, adaptation, réseau de neurones, perceptron multicouche, algorithme de rétro-propagation du gradient, apprentissage.

ABSTRACT

An autonomous navigation system or mobile robot must be able to perceive its environment, build and run a trajectory in appropriate response to the perceived information. That requires a robust heading and speed control system.

This paper presents a study on the control of the direction and speed of an autonomous navigation system using a multilayer perceptron trained with the algorithm of back-propagation of gradient.

The objective is to evaluate the ability of this type of neural network to directly control the heading and speed to be adopted by a mobile robot based on its perceptual inputs to avoid colliding with obstacles in its vicinity.

To achieve this, a strategy of perception and motion planning has been developed and its ability to control an autonomous mobile robot has been addressed. This strategy uses two neural networks operating in parallel , one to control the heading and other for controlling the speed both using the same perceptual system.

Matlab simulation results are presented and analyzed with regard to what machine learning is and what we can expect from it in the context of autonomous navigation.

This approach has the advantage of being simple and indicate not only whether the robot should turn right or left but also how much the steering angle should be unlike the usual approaches described in the literature .

Keywords : teleoperation, autonomous navigation , adaptation, neural network, multilayer perceptron, backpropagation algorithm, machine learning.

INTRODUCTION

Les systèmes de navigation autonomes ou robots mobiles autonomes sont des robots autonomes à base mobile par opposition aux robots manipulateurs autonomes. Ils peuvent être terrestres (voitures autonomes), marins ou sous-marins, (véhicules sous-marins de recherche), aériens (drones). Ce sont des systèmes intelligents qui doivent être capables de percevoir leur environnement, générer et suivre une trajectoire en réaction appropriée à l'information perçue.

Un des premiers objets télé-opérés a été la torpille de Brennan (1877). Elle offrait la possibilité à un opérateur de contrôler, sa direction, jusqu'à une distance de 1,8 km, avec une vitesse pouvant atteindre 50 km/h. Les bases de ce système étaient d'origine mécanique, les commandes de l'opérateur étant transmises par l'intermédiaire des fils. Les origines de la téléopération moderne peuvent êtres considérées comme datant de la fin du 19ème siècle, en étroite liaison avec le développement des communications radio. En 1898, Nikola Tesla décrit la télécommande sans fils d'un bateau miniature. À cette époque (début du vingtième siècle), l'opérateur était en contact visuel direct avec la partie esclave du système. Actuellement, on a la possibilité de télé-opérer sur des distances très importantes. Les sondes spatiales Voyager 1 et 2 sont télé-opérées depuis la terre alors qu'elles se trouvaient à environ 15 milliard kilomètres de celle-ci.

Au début du 21ème siècle, la téléopération trouve ses applications pratiques le plus souvent dans les tâches dites DDD (Dull, Dangerous and Dirty – monotones, dangereuses et sales). Dans les applications aux tâches dangereuses, la robotique spatiale se remarque par les moyens mis en jeu depuis un bon nombre d'années. Le robot Lunokhod 1 a été déposé sur la Lune en 1970 et télé-opéré depuis la Terre pendant 322 jours. On peut également rappeler les rovers martiens Spirit et Opportunity et Sojourner. D'autres domaines dangereux sont les réparations dans des milieux hostiles (l'espace, les profondeurs des océans, les centrales nucléaires), les opérations de déminage, les opérations de recherche et sauvetage (à l'intérieur des immeubles en ruines à la suite d'un tremblement de terre ou une explosion par exemple).

La tendance générale est, à cause des problèmes de fiabilité des communications sans fil, de s'éloigner d'une télé-opération pure (basée sur une philosophie maître-esclave, où la partie esclave n'est munie d'aucune intelligence et exécute « à la lettre » chaque ordre reçu) et de se diriger vers un système coopératif homme-machine, où la machine est munie d'un certain niveau d'intelligence qui lui procure une certaine autonomie.

Les applications potentielles de robotique mobile semi-autonomes ou autonomes sont nombreuses: transport intelligent avec la célèbre voiture autonome de Google (Google Car), recherche et sauvetage, lutte contre les incendie, cartographie, machines minières autonomes, machines agricoles autonomes, applications militaires (reconnaissance et combat avec les drones MQ-1 Predator de l'armée américaine), robotique de service (l'aspirateur automatique Roomba de iRobot).

Pourtant les problèmes à résoudre restent nombreuses et pour la plupart ouverts: structure mécanique, motorisation, alimentation, architecture matérielle et logicielle du système de contrôle/commande.

Les choix de structure sont souvent effectués parmi un panel de solutions connues et pour lesquelles les problèmes de modélisation, planification et commande sont résolus. De même les choix d'actionneurs et d'alimentation sont souvent classiques: moteurs électriques à courant continu avec ou sans collecteur, alimentés par des convertisseurs de puissance fonctionnant sur batterie. Généralement aussi les architectures de contrôle-commande des robots mobiles sont similaires à celles des systèmes automatiques ou robotiques habituelles. Néanmoins on distingue deux niveaux de spécialisation, propres aux systèmes autonomes :

1. couche décisionnelle (planification et gestion des évènements)
2. couche fonctionnelle (génération en temps-réel des commandes des actionneurs)

La couche décisionnelle indique les objectifs de cap et de vitesse tandis que la couche fonctionnelle s'occupe de les réaliser. Les architectures de contrôle définissent comment ces couches devraient être intégrées pour assurer le contrôle latéral et longitudinal du véhicule. Le problème fondamental de la robotique mobile est en effet la génération de trajectoire sans collision c'est-à-dire l'adaptation des directions et vitesses successives du robot aux positions relatives des obstacles présents dans son environnement. Il faut à chaque instant définir la direction suivante à prendre et la vitesse suivante à adopter en fonction de la route et de la configuration des obstacles dans le voisinage du robot.

Ainsi un système de navigation autonome doit accomplir deux fonctions: le guidage latéral (contrôle du cap) et le guidage longitudinal (contrôle de la vitesse).

A chaque itération, un module de guidage latéral définit la prochaine direction à suivre pendant qu'un module de guidage longitudinal spécifie la prochaine vitesse à adopter en fonction de l'objectif visé et des positions relatives des obstacles présents dans l'environnement de déplacement du robot.

Pour résoudre le problème, de nombreux outils sont utilisés au nombre desquels les réseaux de neurones. Ensembles d'opérateurs non linéaires interconnectés, les réseaux de neurones sont en théorie toujours capable de découvrir à partir de quelques exemples bien choisis le modèle sous-jacent à la relation entre

les entrées et les sorties d'un processus sans nécessairement connaitre le mécanisme qui le gouverne. Ce sont des outils adaptés à la réalisation des modèles de conduite; qui visent à établir une approximation réaliste de la corrélation entre les entrées et les sorties d'un processus dont on ne peut pas construire un modèle de connaissance.

L'objectif de ce travail est d'évaluer la capacité d'un perceptron multicouche entrainé avec l'algorithme de rétro-propagation du gradient à calculer directement le cap et la vitesse que doit adopter un robot mobile en fonction de ses entrées perceptives pour éviter d'entrer en collision avec les obstacles présents dans son voisinage.

En effet, dans le cadre de la navigation autonome, les réseaux de neurones servent à faire un couplage direct entre perception et action, un procédé appelé «navigation réactive» et la démarche habituelle pour le faire consiste à se contenter uniquement de dire à chaque itération si le robot doit tourner à droite ou à gauche sans préciser de combien [15]. On ajoute pour compléter, une logique supplémentaire qui exécute un angle de braquage constant (en général 45° ou 90°) à droite si la décision du contrôleur neural est de tourner à droite et du même angle à gauche si la décision est de tourner à gauche.

Or en réalité, les angles de braquage peuvent prendre n'importe quelle valeur entre 0° et le maximum.

Dans ce travail nous proposons que les angles de braquage comme les vitesses soient variables et que deux réseaux de neurones indépendants les calculent directement et en parallèle.

Ce mémoire est divisé en quatre chapitres. Le premier est destiné à une entrée en matière à la télé-opération. Dans le deuxième chapitre, une classification typologique des différentes architectures de la navigation autonome a été présentée. Le troisième chapitre décrit l'approche que nous avons proposée tandis que le quatrième chapitre montre les résultats et leur interprétation.

CHAPITRE 1

INTRODUCTION À LA NAVIGATION AUTONOME

Dans ce chapitre nous faisons une analyse du processus de télé-opération. Nous présentons une architecture de mobilité classique d'un véhicule télé-opéré et les principales approches utilisées pour sa mise en œuvre.

1.1 Le concept de télé-opération basée sur le retour visuel

La navigation autonome est un cas particulier du concept général de télé-opération basée sur un retour visuel. Le problème consiste à doter le mobile de la capacité de se mouvoir dans n'importe quel environnement (structuré ou non) en se basant uniquement sur des données collectées par ses propres capteurs perceptifs. On parle généralement de commande référencée capteurs pour qualifier ce type de traitement, qui se résume à l'utilisation d'une correspondance entre l'espace des données capteurs et l'espace des commandes du robot qui permet de savoir à chaque instant la configuration de son voisinage pour décider de la direction et de la vitesse à adopter à l'instant suivant. Le système de perception est toujours intégré au véhicule alors que le guidage peut être opéré à distance. Le robot peut ainsi soit être guidé à distance à partir d'un centre de contrôle ou il peut s'autoguider avec un module de contrôle interne. Dans le premier cas, on parle de télé-opération, alors que dans le second, on parle de navigation autonome.

1.2 Architecture de mobilité d'un véhicule automatisé

La chaîne fonctionnelle d'un véhicule automatisé, représentée sur la figure 1-1, fait traditionnellement intervenir trois modules de commande organisés hiérarchiquement [30]:

- le module de ***planification*** calcule un itinéraire local sous forme de points de passage à partir d'une destination définie par un opérateur humain et des informations de localisation (positions, cap) qu'il recueille sur le véhicule au moyen de capteurs proprioceptifs. Il peut calculer également une vitesse moyenne ou un profil de vitesse de consigne associé à cet itinéraire, en fonction du délai spécifié par l'opérateur pour la réalisation de la tâche et/ou de limitations de vitesse exigée par la loi. *Ce module spécifie la route*. C'est l'une des fonctions assurées par le conducteur humain.

- le module de ***guidage*** calcule les consignes de vitesse et les consignes d'orientation permettant au centre de gravité du véhicule (ou à tout autre point du véhicule, appelé *point de commande*) de suivre une trajectoire. Concrètement ce module interpole les points de passage calculés par le module de planification relativement aux positions relatives des obstacles présents dans l'environnement de navigation. *Ce module spécifie les directions à suivre*. C'est aussi une fonction dévolue au conducteur humain.
- le module de ***pilotage*** a pour fonction d'asservir la vitesse et le cap du point de commande sur les valeurs de consigne déterminées par le module de guidage, c'est-à-dire d'élaborer les commandes à délivrer aux actionneurs de vitesse et de direction du véhicule. *Ce module définit comment suivre les directions et les vitesses de consigne*. Cette fonction est réalisée par un asservissement interne au véhicule.

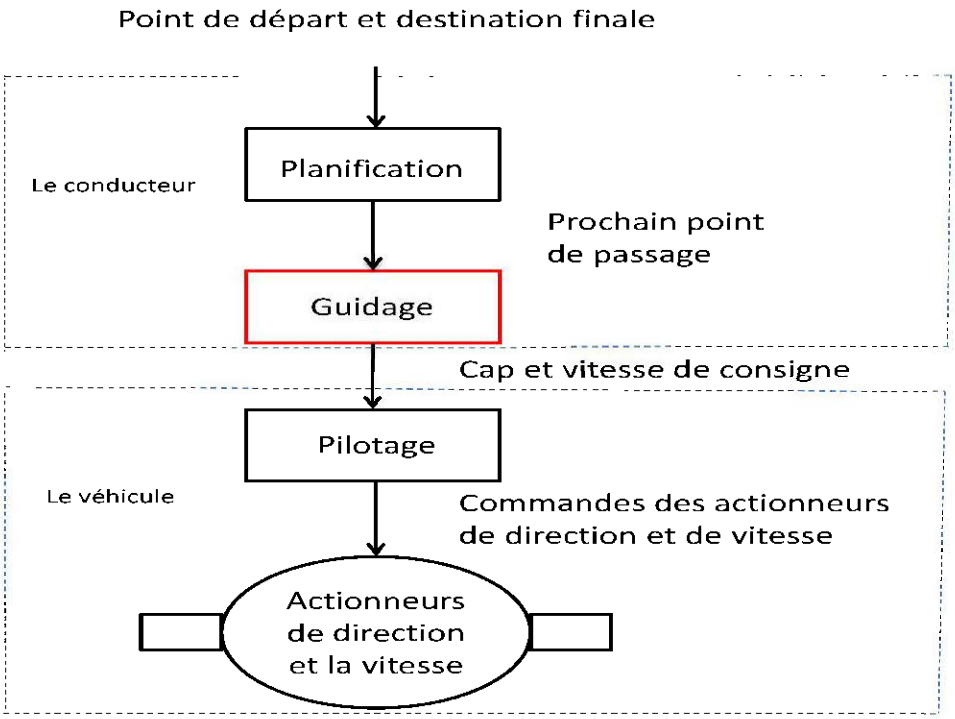


Figure 1- 1: Architecture de mobilité classique d'un véhicule automatisé

Dans ce travail, nous traiterons du guidage. Le système à mettre en œuvre ici se contentera d'indiquer à chaque itération la direction et la vitesse à adopter sans se préoccuper des questions relatives à la

définition de l'itinéraire (tâche du module de planification) ni de comment la consigne de direction ou de vitesse va être réalisée (tâche du module de pilotage).

1.3 Niveaux d'intervention

On donne une dénomination différente au mode de commande du véhicule automatisé selon le niveau où intervient l'opérateur humain [30]. On distingue quatre modes de commande: le *télépilotage*, le *téléguidage*, la *téléplanification* et la *navigation autonome*. Les trois premiers modes de commande sont de la télé-opération.

1.3.1 Télépilotage

On parle de *télépilotage* (ou *téléconduite*) lorsque les ordres, ou consignes, qui sont transmis au véhicule par l'opérateur humain, sont les commandes des actionneurs. Ce mode requière la présence d'un opérateur humain pour commander les actionneurs du véhicule. L'opérateur doit disposer d'informations sensorielles (visuelles surtout) représentatives de celles qu'il aurait s'il se trouvait à bord du véhicule. C'est le mode de commande où l'autonomie du véhicule est la plus limitée. Il est relativement sûr, puisque proche d'une conduite humaine, mais n'est véritablement praticable qu'à basse vitesse sur terrain peu accidenté. En effet, cette approche nécessite une infrastructure de communication haut débit et quasi-temps réel. Il n'est pas possible au-delà de 30 km/h de négliger le temps de transfert des données entre l'opérateur et le véhicule dans le réseau de communication : au delà de 30 km/h, un système télé-piloté devient en général instable [30]. De plus, les informations sensorielles retournées à l'opérateur sont le plus souvent insuffisantes pour une conduite correcte en tout terrain (champ de vision limité, insuffisance du retour d'informations concernant d'autres données sensorielles, telles que les sons, les vibrations, les accélérations…).

1.3.2 Téléguidage

Il s'agit de *téléguidage*, lorsque les consignes communiquées au véhicule sont des consignes de vitesse et de cap. La fonction de pilotage est dans ce cas assurée par le véhicule lui-même. Un tel mode de commande ne souffre pas des inconvénients du télé-pilotage. L'autonomie du véhicule y est cependant encore restreinte, puisque la définition de la trajectoire à suivre incombe à l'opérateur. Pour que cette trajectoire soit optimale, il est nécessaire que l'opérateur ait une bonne connaissance de la dynamique du véhicule.

1.3.3 Téléplanification

On parle de *téléplanification*, lorsque le véhicule reçoit comme consignes une liste de points de passage assortie d'une contrainte temporelle. Les fonctions de guidage et de pilotage sont assurées au niveau du véhicule. Ici le véhicule détermine lui-même sa trajectoire à partir des consignes de passage qu'il reçoit de l'opérateur humain et des positions des obstacles dans son environnement. Le véhicule a plus de pouvoir de décision.

1.3.4 L'autonomie complète

L'autonomie est effective, si toutes les fonctions de planification, guidage et pilotage sont assurées au sein du véhicule. En présence d'obstacles, localisés par un système de détection d'obstacles, le module de guidage embarqué redéfinira lui-même les points de passage en s'appuyant sur les informations de localisation du véhicule et le chemin que lui donne le module de planification. Cette approche a l'avantage de favoriser la réduction du volume d'informations échangées entre l'opérateur et le système. On délègue une partie importante des fonctions de navigation au système à piloter; ce qui permet de s'affranchir des contraintes de bande passante et de délais. Cependant, elle exige la mise en œuvre d'un asservissement local intelligent.

1.4 Problématique de la navigation autonome

Nous tenons d'abord à préciser que la question de guidage contient en fait deux problèmes: le guidage latéral (contrôle du cap) et le guidage longitudinal (contrôle de la vitesse).

Soit M le mobile à guider, se déplaçant dans un espace euclidien W (espace de travail) de dimension 2 ou 3. Soient O_1, O_2,...,O_q des objets rigides, obstacles dispersés dans W.

Le problème du guidage latéral se formule comme suit: étant donné l'itinéraire à suivre pour arriver à destination, il faut générer une trajectoire C spécifiant la séquence des directions successives que doit prendre le robot relativement aux obstacles pouvant se présenter sur son chemin pour atteindre la destination sans entrer en collision avec ces derniers.

Quant au problème du guidage longitudinal, il se formule comme suit: étant donné à chaque instant la position du robot et celle de la destination, il faut générer le profil de vitesse en spécifiant les consignes de vitesses à adopter pour:

- prendre les virages,
- éviter la collision avec les obstacles

- et s'arrêter à destination.

On peut fusionner les solutions au contrôle latéral et longitudinal en une ou les séparer en deux entités indépendantes évoluant en parallèle.

Une analyse du processus de guidage permet de décomposer le module de guidage en deux sous-modules: perception et décision comme le montre la figure 1-2. Considérons un espace de travail de dimension 2 et un référentiel fixé au robot comme le montre la figure 1-3. A chaque instant, le prochain point de passage est donné par un module qui spécifie la route (planificateur de route) et les positions des obstacles relativement au robot sont connues grâce à des détecteurs disposés sur le robot; qui forment ainsi son système de perception. A chaque itération, un module de décision se sert des données perceptives pour déterminer le prochain cap du robot. Aucune historique des lectures des détecteurs d'obstacles n'est retenue de sorte que le robot se déplace de façon purement réactive entre les obstacles. Les obstacles, qui peuvent être fixes ou mobiles.

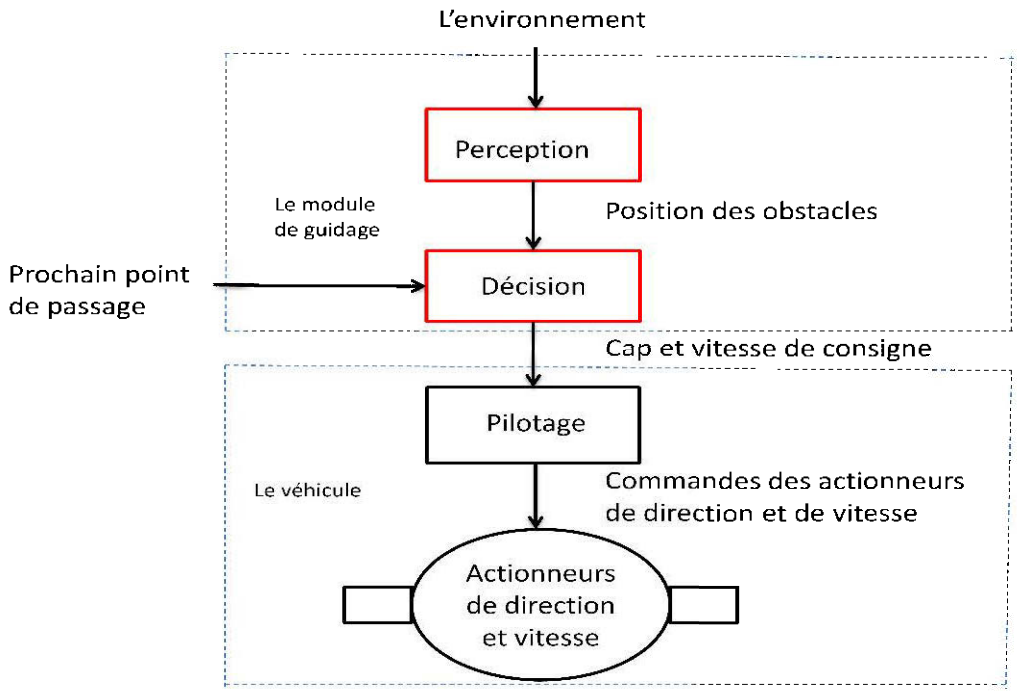


Figure 1- 2: Architecture de guidage classique d'un mobile.

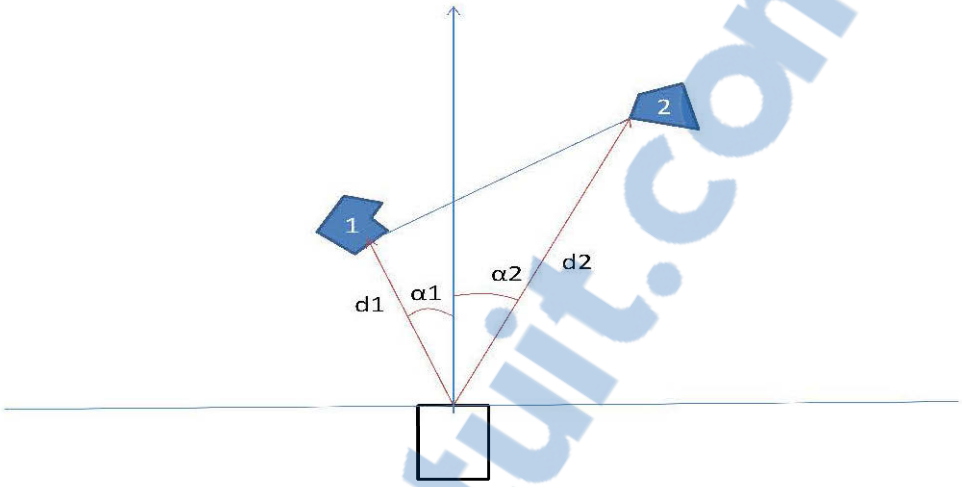


Figure 1- 3: Les obstacles sont localisés relativement au robot

1.5 Les principales approches

On rencontre dans la littérature beaucoup de stratégies et outils pour la commande autonome de robots mobiles. Ces travaux se sont historiquement séparés en trois groupes dont les deux premiers se sont longtemps opposés [26].

1.5.1 Les architectures délibératives

L'approche délibérative utilise une modélisation de l'environnement, connue à priori ou obtenue à partir des données capteurs, pour planifier à l'avance les commandes que le robot doit exécuter. Cette vision de la robotique autonome conduit à une décomposition séquentielle du traitement réalisé et à des systèmes fortement hiérarchiques.

Cette approche prend ses origines des techniques de l'intelligence artificielle classique. Elles visent à reproduire une certaine vision de la démarche humaine de résolution de problème. Le traitement est décomposé en une série d'opérations successives comme décrites par la figure 1-4.

La première opération consiste à traiter les données sensorielles qui fournissent au robot des informations sur son environnement (perception).

L'étape suivante consiste à construire ou à mettre à jour, à partir des données capteurs, un modèle du monde dans lequel évolue le robot. Ce modèle peut être par exemple une identification du type et de la position des obstacles présents sur le chemin du robot. Ce modèle est utilisé pour planifier une suite d'états conduisant le robot à effectuer la tâche recherchée (planification).

La dernière opération a pour but de calculer puis d'exécuter les actions permettant au robot de suivre le plan généré par l'étape précédente (action). Après chaque mouvement, le robot s'arrête et met à jour les informations lui permettant d'exécuter le mouvement suivant. Puis il répète le processus jusqu'à l'atteinte de la destination. C'est une approche du haut vers le bas communément appelé top-down où la tâche globale de haut niveau est décomposée en une suite de tâches élémentaires de bas niveau dont la réalisation successive aboutit à l'atteinte de l'objectif global.

Cependant, la construction d'un modèle de l'environnement et d'un plan est une démarche coûteuse en temps de calcul particulièrement en environnement non-structuré. Cela fait que l'intervalle de temps entre la perception et l'action peut être long. Ce qui n'est guère adapté à un processus temps réel comme la navigation. Car souvent le robot n'a pas encore fini de générer ou d'exécuter la commande lui permettant d'éviter une collision que déjà la collision se produit. Aussi, dans une architecture délibérative, on fait les hypothèses fortes que le modèle de l'environnement utilisé est précis et que le planificateur ne se trompe pas ; ce qui n'est pas non plus réaliste. Enfin dans une telle architecture, le fonctionnement global du système dépend du fonctionnement en particulier de chaque constituant. De sorte que si un module venait à tomber en panne, le système au complet s'écroulerait [26].

Néanmoins les systèmes délibératifs possèdent un point fort important: leur capacité à prendre en compte des raisonnements de haut niveau lors de la phase de planification. Il leur est possible en effet de gérer des missions complexes comprenant plusieurs buts successifs [26]. Ce qui est comme nous le verrons dans la section suivante, chose impossible pour les architectures purement réactives.

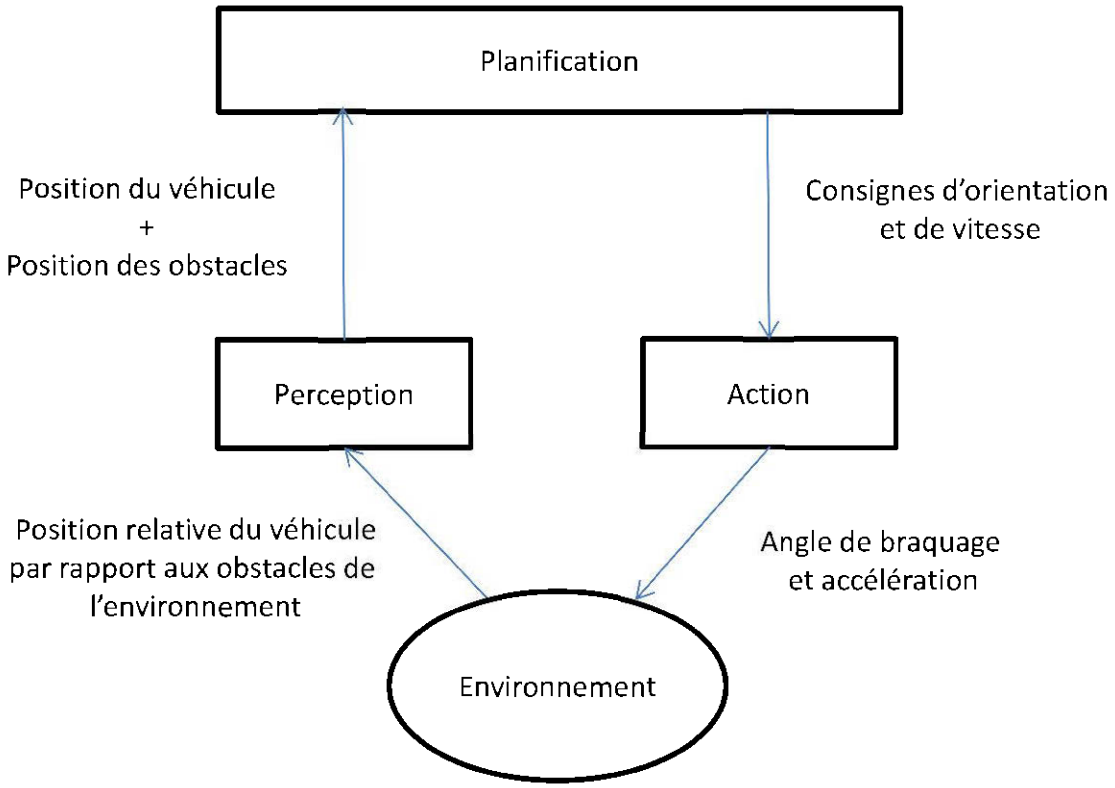


Figure 1- 4: Architecture délibérative

1.5.2 Les architectures réactives

L'approche réactive, s'appuie sur un couplage étroit entre les capteurs et les actionneurs, pour générer en continu les commandes de navigation. Cette méthode va généralement de pair avec une décomposition sous forme de comportements élémentaires réalisés successivement.

La navigation réactive ou orientée-comportement a été proposée la première fois par J. Brooks dans un article publié en 1986 en réaction aux insuffisances des modèles délibératifs dans les environnements dynamiques et inconnus [4]. Cette approche, radicalement différente de l'approche délibérative, se distingue par l'abandon des phases de modélisation et de planification. Elle est née à partir de considérations issues de l'éthologie et plus particulièrement de l'entomologie [1]. En effet, si on constate que de nombreux êtres vivants, dont en particulier les insectes sont capables en dépit de leur faible faculté de modélisation et de raisonnement d'accomplir des tâches complexes résultant d'une intelligence avancée, similaire par certains côtés à la notre, on peut se demander si la modélisation de l'environnement

et la planification du mouvement sont aussi importantes en navigation que l'exigent les architectures délibératives [4].

Des chercheurs sont partis de cette constatation pour proposer une architecture de commande nouvelle basée sur la composition de plusieurs modules implémentant des comportements simples. Le comportement global du robot est alors le résultat de la composition des divers comportements élémentaires. Il émerge de l'interaction de ces derniers entre eux et avec l'environnement. Chacun des comportements de base, qui peuvent être par exemple l'évitement d'obstacle, ou la poursuite d'un objet, réalise un couplage étroit entre les capteurs et les actionneurs du robot en générant les commandes sur la base d'informations courantes et locales. Chaque action est une réaction instantanée à une information reçue à un moment donné sans aucune commune relation avec une planification à long terme. La simplicité du traitement ainsi réalisé, de même que le caractère parallèle des actions, permettent une grande réactivité.

La figure 1-5 montre la structure des modèles basés sur cette approche. Dans la première couche, un module de perception apporte les informations sensorielles au robot. Dans la couche suivante, un ou plusieurs modules appelés comportements (ou actions) fonctionnent en parallèle. Chaque module comportemental reçoit des inputs sensoriels propres à sa fonction et produit la réponse appropriée correspondante. Finalement le robot adopte l'action dictée par la sortie du comportement actif. Ainsi des tâches de navigation complexes sont segmentées en de nombreuses sous-tâches élémentaires qui contribuent chacune à améliorer la performance globale du système.

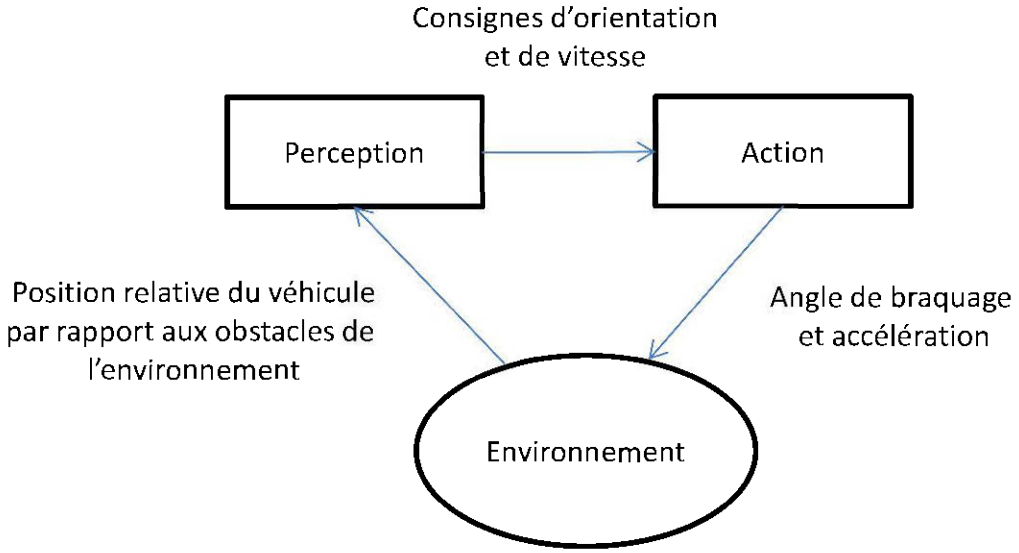


Figure 1- 5: Architecture réactive

1.5.3 Les architectures hybrides

L'approche hybride tente de combiner les deux approches précédentes afin de tirer partie des avantages respectifs de ces deux techniques. Ce dernier groupe représente aujourd'hui la grande majorité des systèmes étudiés (figure 1-6).

Pour pallier à la fois aux insuffisances des systèmes délibératifs et réactifs, certains chercheurs ont tenté de combiner des propriétés des deux approches pour obtenir une architecture mixte qu'on désigne par architecture hybride. Le but de leur démarche est de conserver les capacités de raisonnement de haut niveau des approches délibératives, tout en s'assurant la robustesse et la rapidité de réponse des approches réactives. Les solutions adoptées pour combiner réaction et délibération sont très variées. Indépendamment du procédé utilisé, Le module délibératif (planification) s'occupe des traitements de haut niveau (fusion de capteurs, construction de carte, planification etc.). Ses sorties sont envoyées au module réactif (action) qui les utilise pour générer les mouvements du robot. Ainsi, l'intégration de différentes propriétés dans l'architecture hybride conduit à une nouvelle architecture de navigation, flexible et robuste. Ici, l'action dépend de la perception et/ou de la planification selon la situation. En effet, en absence d'obstacle les consignes de la planification ont la priorité. Alors qu'en présence d'obstacle, les consignes de la perception ont la priorité. Il n'y a pas de risque d'auto-blocage à suivre deux ordres parallèles. A cet effet, il suffit de s'imaginer ce qui se passe lorsque nous dévions de la route indiquée par notre navigateur GPS. Le calcul d'un nouvel itinéraire se met immédiatement en branle et se charge de nous ramener sur le chemin de la destination.

L'architecture que nous proposons dans ce travail, au chapitre 3, est de type hybride.

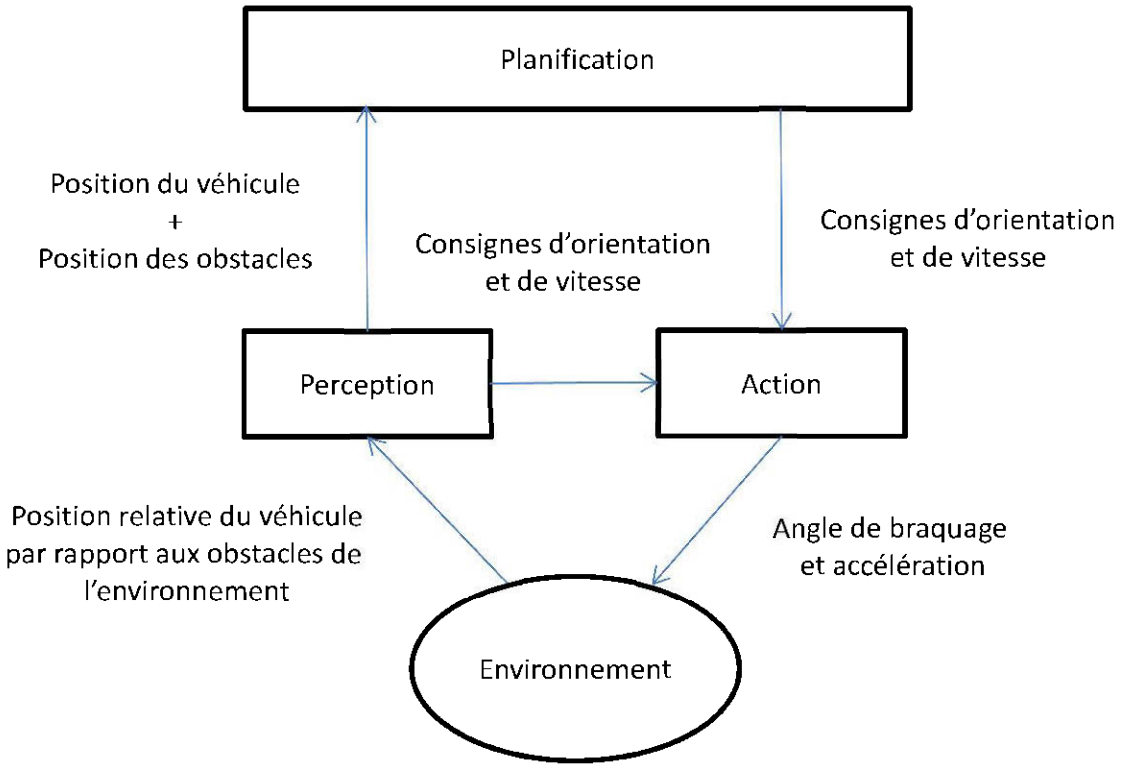


Figure 1- 6: Architecture hybride

1.6 Analyse comparative des approches

De nombreux critères permettent de caractériser les différentes approches. Les critères les plus souvent utilisés sont:

- l'orientation-objectif c'est-à-dire la capacité du système de contrôle à atteindre des objectifs multiples
- la flexibilité: possibilité d'ajouter de nouveaux modules en vue de réaliser des tâches supplémentaires
- la facilité de mise en œuvre. Elle réfère à la facilité relative de comprendre l'architecture, la développer, la tester et la debugger
- la réactivité c'est-à-dire l'habileté du système à répondre et s'adapter aux variations soudaines de l'environnement
- l'optimalité de l'exécution. Elle réfère à la précision des manœuvres
- la capacité d'apprentissage: l'habileté du système à apprendre
- la robustesse c'est-à-dire la capacité du système à accommoder les changements soudains, les entrées incomplètes ou imprécises, les dysfonctionnements etc.
- la planification: la capacité à « voir » sur le long terme

- l'efficacité: la capacité du système à tirer le meilleur de chaque comportement et d'intégrer tous les comportements pour générer une trajectoire douce.

Avec les descriptions que nous avons données des différentes approches aux paragraphes précédents, on peut dire que les architectures délibératives sont bonnes en terme d'orientation-objectif, d'optimalité de l'exécution, de capacité d'apprentissage et de planification. Elles ont cependant une moins bonne performance pour les autre critères. Les architectures réactives sont flexibles, faciles à mettre en œuvre, réactives et efficaces. Elles sont par contre peu précises, peu orientées-objectif et ont une moindre capacité d'apprentissage. Les architectures hybrides quant à elles réalisent une synthèse des deux précédentes en offrant une bonne orientation-objectif, une bonne flexibilité et facilité de mise en œuvre, une très bonne réactivité et efficacité. Elles sont par contre peu performantes pour gérer le long terme et ont une capacité d'apprentissage modérée. Les tableaux 1-1 et 1-2 résument les comparaisons.

Les réseaux de neurones, en particulier les perceptrons multicouches que nous présenterons au chapitre suivant nous serviront à mettre en œuvre une architecture de type hybride.

Caractéristique de l'architecture	**Délibérative**	**Réactive**	**Hybride**
Orientation-objectif	Très bonne	Pas bonne	Bonne
Flexibilité	Très mauvaise	Très bonne	Très bonne
Facilite de mise en œuvre	Très mauvaise	Très bonne	Bonne
Réactivité	Très mauvaise	Très bonne	Bonne
Optimalité de l'exécution	Très bonne	Très mauvaise	Bonne
Capacité d'apprentissage	Très bonne	Modérée	Modérée
Robustesse	Pas bonne	Bonne	Très bonne
Planification	Très bonne	Pas bonne	Bonne
Efficacité	Pas bonne	Très bonne	Très bonne

Tableau 1- 1: Comparaison des architectures de contrôle

Caractéristiques	Description
Orientation-objectif	Capacité du système de contrôle à atteindre des objectifs multiples
Flexibilité	Possibilité d'ajouter de nouveaux modules en vue de réaliser des tâches supplémentaires
Facilite de mise en œuvre	Réfère a la facilite relative de comprendre l'architecture, la développer, la tester et la debugger
Réactivité	Habileté du système à répondre et s'adapter aux variations soudaines de l'environnement
Optimalité de l'exécution	Réfère à la précision des manœuvres
Capacité d'apprentissage	Habileté du système à apprendre
Robustesse	Capacité du système à accommoder les changements soudains, les inputs incomplets ou imprécis, les dysfonctionnements etc.
Planification	Capacité à « voir » sur le long terme
Efficacité	Capacité du système à tirer le meilleur de chaque comportement et d'intégrer tous les comportements pour générer une trajectoire douce.

Tableau 1- 2: Définition des critères de comparaison des architectures

CHAPITRE 2

LES RÉSEAUX DE NEURONES

2.1 Les architecture de réseaux de neurones

Dans le chapitre précédent nous avons exposé le problème de la navigation autonome. Dans ce chapitre-ci nous décrirons l'outil que nous utiliserons dans notre proposition au chapitre 3.

2.1.1 Le parallèle réseau de neurones et cerveau

Un réseau de neurones artificiels est un ensemble fortement interconnectés de processeurs élémentaires (dits neurones artificiels) fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Le fonctionnement d'un neurone artificiel est inspiré du celui d'un neurone biologique.

Le cerveau est un système de reconnaissance constitué d'un amas d'unités de reconnaissance (neurones). Le cerveau est en effet tout le temps en train de reconnaitre quelque chose ; un visage, un bruit, un objet, une odeur, une mélodie etc. Le fait est que la réalité ne nous arrive pas à l'esprit d'un trait. Le cerveau reçoit les signaux par morceaux et doit construire l'entité à reconnaitre avec ces morceaux en faisant des liens entre eux puis en vérifiant si l'ensemble obtenu correspond à la réalité. Quand le cerveau est face à une scène, chaque neurone reçoit plusieurs entrées en provenance de la scène (des morceaux de la scène), établit des liens entre ces entrées (liens dentitriques) puis « lève un drapeau » si sa construction s'apparente suffisamment à la scène perçue (émission de potentiel d'action).

En 1943, deux neurobiologistes américains J. Mc Culloch et W. Pitts ont eu l'idée de modéliser mathématiquement cette propriété. Au regroupement des entrées par mise en relation entre elles, ils ont associé une sommation pondérée des entrées. Et à l'émission de potentiel d'action, ils ont associé l'application d'une fonction non-linéaire à la somme pondérée [28].

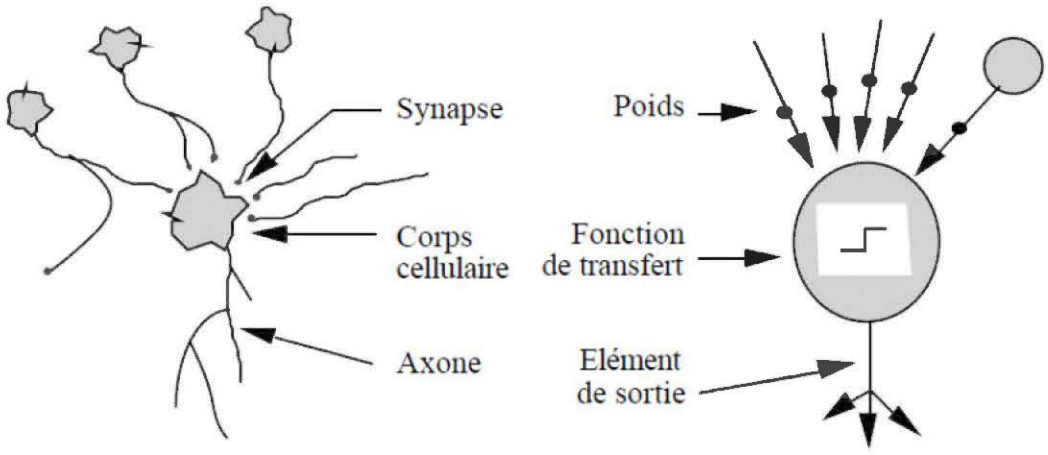


Figure 2- 1: Mise en correspondance neurone biologique/neurone artificiel

Le neurone biologique possède trois principales composantes : les *dendrites*, le *corps cellulaire* et l'*axone*. Les dendrites forment un maillage de récepteurs nerveux qui permettent d'acheminer vers le corps du neurone des signaux électriques en provenance d'autres neurones. Celui-ci agit comme une espèce d'intégrateur assemblant les signaux reçus. Il accumule ainsi des charges électriques. Et lorsque la charge accumulée dépasse un certain seuil, il émet un potentiel électrique qui se propage à travers l'axone pour éventuellement venir exciter d'autres neurones.

Comme nous l'avons dit plutôt, c'est en se basant sur ces connaissances biologiques que le modèle mathématique a été défini. Comme le neurone biologique, le neurone artificiel possède aussi trois fonctions : la *fonction de sommation pondérée*, la *fonction de transfert* et la *fonction de comparaison*. Il reçoit un nombre variable d'entrées en provenance des neurones amont. A chacune de ces entrées est associé un poids w puis les entrées pondérées sont sommées. La quantité obtenue passe dans une fonction de transfert. Il en résulte une autre quantité qui est comparée à une valeur seuil. Dépendamment du résultat de cette comparaison, une valeur décision est émise et transmise aux neurones avals.

On peut résumer le parallèle entre le neurone biologique et le neurone artificiel par le tableau 2-1, qui nous permet de voir clairement la correspondance entre les deux objets [28].

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

Tableau 2- 1: Analogie entre neurone biologique et neurone artificiel

2.1.2 Modèle d'un neurone

Le modèle mathématique d'un neurone artificiel est illustré à la figure 2-2. Un neurone est donc essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées. Le résultat n de cette somme est ensuite transformée par une fonction de transfert f qui produit la sortie a du neurone.

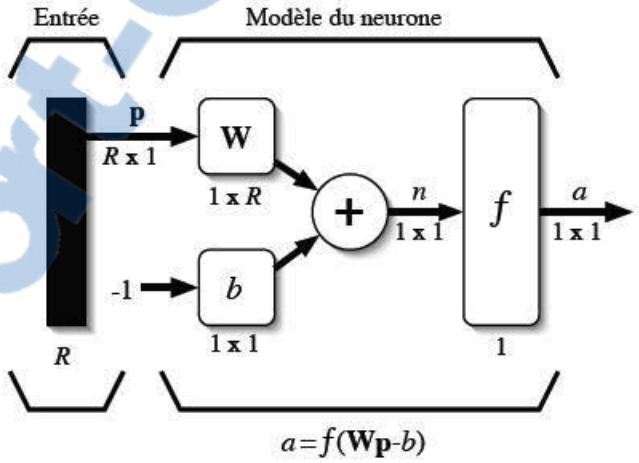


Figure 2- 2: Représentation matricielle du modèle d'un neurone artificiel

Les R entrées du neurone correspondent au vecteur $p = [p_1\ p_2 \cdots p_R]^T$, alors que $w = [w_{1,1}\ w_{1,2} \cdots w_{1,R}]^T$ représente le vecteur des poids du neurone. La sortie n de l'intégrateur est donnée par l'équation suivante :

$$\begin{aligned} n &= \sum_{j=1}^{R} w_{1,j}\, p_j - b \\ &= w_{1,1}\, p_1 + w_{1,2}\, p_2 + \cdots + w_{1,R}\, p_R - b, \end{aligned} \tag{2.1}$$

que l'on peut aussi écrire sous forme matricielle :

$$n = \mathbf{w}^T \mathbf{p} - b. \tag{2.2}$$

Cette sortie correspond à une somme des entrées pondérée des poids moins ce qu'on nomme le biais b du neurone. Le résultat n de la somme pondérée s'appelle le niveau d'activation du neurone.

Le biais b s'appelle aussi le seuil d'activation du neurone. Dans le cas de la fonction de transfert seuil, lorsque le niveau d'activation atteint ou dépasse le seuil, l'argument de la fonction de transfert f est positif (ou nul). Dans le cas contraire, il est négatif. Mais en règle générale on a :

$$a = f(n) = f(\mathbf{w}^T \mathbf{p} - b). \tag{2.3}$$

ou

$$a = f(\mathbf{W}\mathbf{p} - b). \tag{2.4}$$

Si on remplace $\mathbf{w}^T$ par la matrice uniligne $\mathbf{W}$.

Il existe de nombreuses formes possibles pour la fonction de transfert. Les différentes fonctions de transfert pouvant servir de fonction d'activation de neurone sont énumérées au tableau 2-2 [28]. Les plus utilisées sont les fonctions «seuil», «linéaire» et «sigmoïde».

Nom de la fonction	Relation d'entrée/sortie	Icône	Nom Matlab
seuil	$a = 0 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$		hardlim
seuil symétrique	$a = -1 \quad \text{si } n < 0$ $a = 1 \quad \text{si } n \geq 0$		hardlims
linéaire	$a = n$		purelin
linéaire saturée	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } 0 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$		satlin
linéaire saturée symétrique	$a = -1 \quad \text{si } n < -1$ $a = n \quad \text{si } -1 \leq n \leq 1$ $a = 1 \quad \text{si } n > 1$		satlins
linéaire positive	$a = 0 \quad \text{si } n < 0$ $a = n \quad \text{si } n \geq 0$		poslin
sigmoïde	$a = \frac{1}{1+\exp^{-n}}$		logsig
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
compétitive	$a = 1 \quad \text{si } n \text{ maximum}$ $a = 0 \quad \text{autrement}$		compet

Tableau 2- 2: Les principales fonctions de transfert

Comme son nom l'indique, la fonction seuil applique un seuil sur son entrée. Plus précisément, une entrée négative ne passe pas le seuil, la fonction retourne alors la valeur 0 (on peut interpréter ce 0 comme signifiant faux), alors qu'une entrée non-négative dépasse le seuil, et la fonction retourne 1 (vrai).

Utilisée dans le contexte d'un neurone, cette fonction est illustrée à la figure 2-3 a). On remarque alors que le biais b dans l'expression de $a = \text{hardlim}(\mathbf{w}^T p - b)$ détermine l'emplacement du seuil sur l'axe $\mathbf{w}^T p$, où la fonction passe de 0 à 1. Cette fonction permet de prendre des décisions binaires.

La fonction linéaire est très simple, elle affecte directement son entrée à sa sortie : $a = n$. Appliquée dans le contexte d'un neurone, cette fonction est illustrée à la figure 2-3 b). Dans ce cas, la sortie du neurone correspond à son niveau d'activation dont le passage à zéro se produit lorsque $\mathbf{w}^T p = b$.

La fonction de transfert sigmoïde est quant à elle illustrée à la figure 2-3 c). Son équation est donnée par : $a = 1/ (1+e^{-n})$

Selon que l'on est loin ou près de b, cette fonction ressemble à la fonction seuil ou à la fonction linéaire.

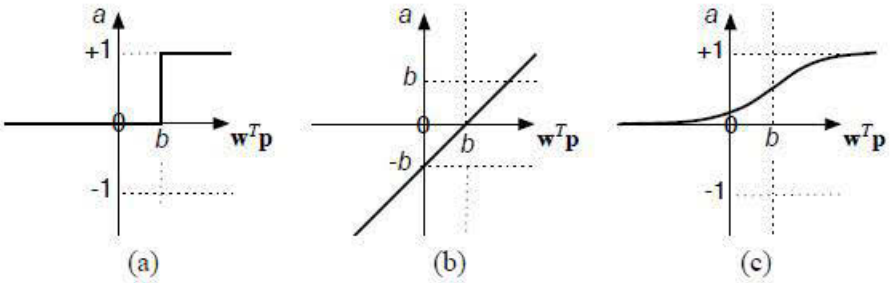


Figure 2- 3: Fonction de transfert : (a) du neurone «seuil» ; (b) du neurone «linéaire», et (c) du neurone « sigmoïde ».

2.1.3 Modèle d'un réseau de neurones

Un réseau de neurones est un maillage de plusieurs neurones, généralement organisé en couches. Pour construire une couche de S neurones, il suffit de les assembler comme à la figure 2-4 [28]. Chacun des S neurones est branché aux R entrées. On dit alors que la couche est totalement connectée. Un poids $w_{i,j}$ est associé à chacune des connexions. Nous noterons le premier indice par i et le deuxième par j. Le premier indice (ligne) désignera le numéro de neurone sur la couche, alors que le deuxième indice (colonne) spécifie le numéro de l'entrée. Ainsi, $w_{i,j}$ désigne le poids de la connexion qui relie le neurone i à son entrée j. L'ensemble des poids d'une couche forme donc une matrice W de dimension S × R :

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \tag{2.5}$$

A noter qu'en général on a : S≠R (les nombres de neurones et d'entrées sont indépendants).

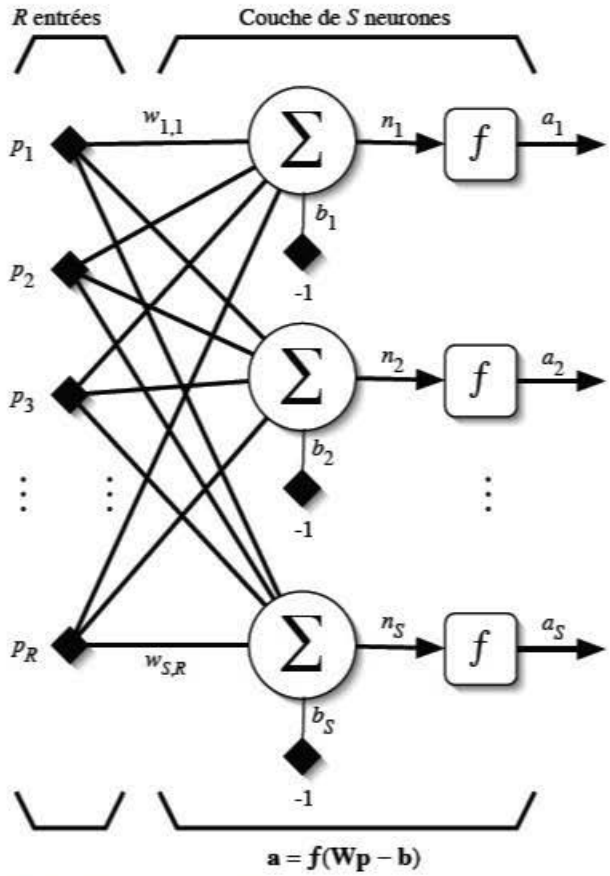


Figure 2- 4: Couche de S neurones totalement connectés

Si l'on considère que les S neurones forment un vecteur de neurones, alors on peut créer les vecteurs $b = [b_1\ b_2 \cdots b_S]^T$, $n = [n_1\ n_2 \cdots n_S]^T$ et $a = [a_1\ a_2 \cdots a_S]^T$. Ceci nous amène à la représentation graphique simplifiée, illustrée à la figure 2-5.

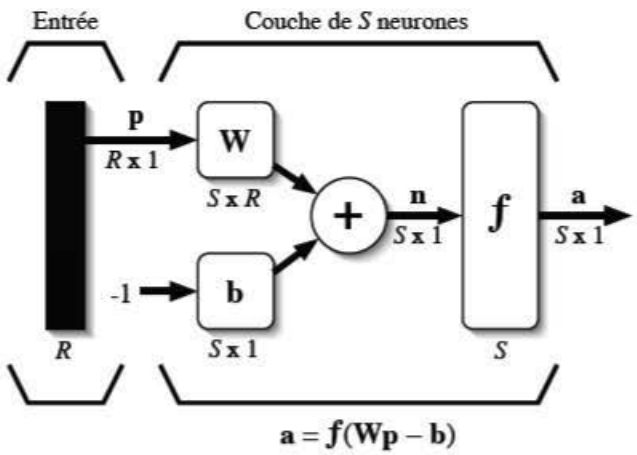


Figure 2- 5: Représentation matricielle d'une couche de S neurones.

Finalement, pour construire un réseau, il ne suffit plus que de combiner des couches comme à la figure 2-6 [28]. Cet exemple comporte R entrées et trois couches de neurones comptant respectivement S1, S2 et S3 neurones. En général encore, on a : S1 ≠S2 ≠ S3. Chaque couche k possède donc son propre nombre de neurones S_k , sa propre matrice de poids W_k, son propre vecteur de biais b_k, son propre vecteur de somme pondérée n_k et son propre vecteur de sortie a_k

Il est important de remarquer que les sorties d'une couche constituent les entrées de la couche suivante. Ainsi, on peut théoriquement empiler autant de couches que l'on veut comme on peut disposer d'autant de neurones que l'on veut. Il faut noter aussi que l'on peut changer de fonction de transfert d'une couche à l'autre. Ainsi, toujours dans le cas général on a : $f^1 \neq f^2 \neq f^3$.

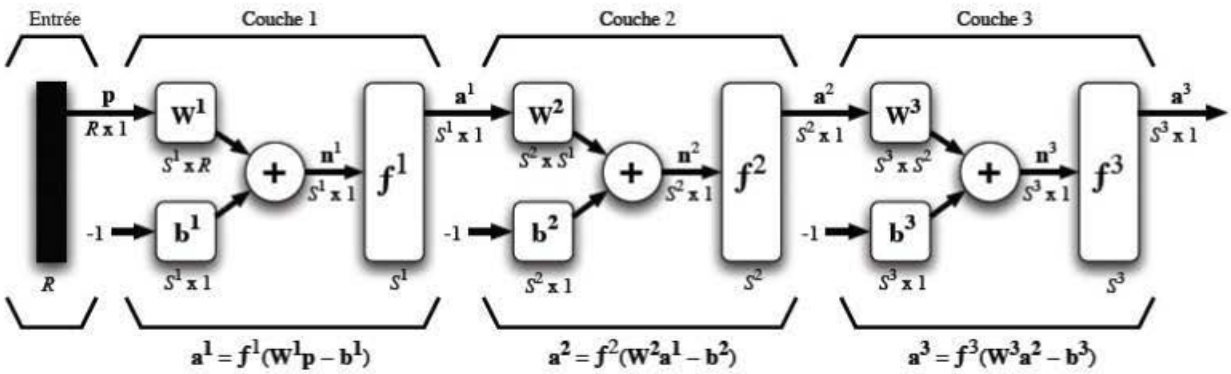


Figure 2- 6: Représentation matricielle d'un réseau de trois couches

La dernière couche est appelée «couche de sortie». Toutes les couches qui précédent la couche de sortie sont appelées «couches cachées». Le réseau de la figure 2-6 possède donc deux couches cachées et une couche de sortie.

2.1.4 Les perceptrons

Les perceptrons sont des réseaux de neurones dans lesquels l'information se propage de couche en couche sans retour en arrière possible. Pour cela, on les appelle aussi réseaux de neurones non-bouclés.

Un perceptron monocouche est illustré à la figure 2-7 [28]. Il s'agit d'une seule couche de S neurones totalement connectés sur un vecteur p de R entrées. La matrice $\mathbf{W} = [w_1\ w_2 \cdots w_S]^T$ de dimension S×R représente l'ensemble des poids de la couche, avec les vecteur-lignes w_i de dimension R×1 représentant les R poids des connexions reliant le neurone i avec ses entrées. Le vecteur b de dimension S×1 désigne l'ensemble des S biais de la couche. Les niveaux d'activation $n = \mathbf{W}p - b = [n_1\ n_2 \cdots n_S]^T$ des neurones de la couche servent d'argument à la fonction d'activation qui applique un seuil au niveau 0 pour produire le vecteur des sorties $a = [a_1\ a_2 \cdots a_S]^T$ où :

$$a_i = \begin{cases} +1 & \text{si } n_i \geq 0 \\ -1 & \text{autrement} \end{cases}$$

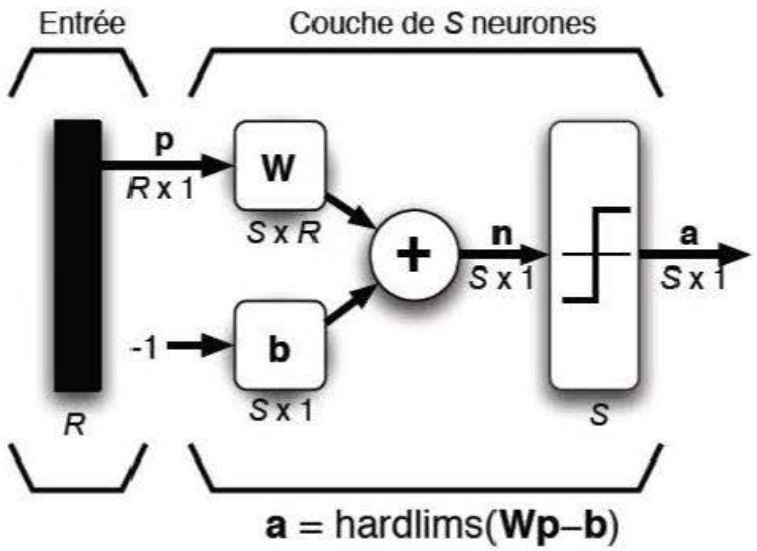


Figure 2- 7: Perceptron à une seule couche avec fonction de transfert seuil

Considérons le cas simple, avec une couche à un neurone relié à deux entrées R = 2 et S = 1,

Dans ce cas, nous aurons $p = [p_1 \ p_2]^T$, $W = [w_1]^T = [w_{1,1} \ w_{1,2}]$, $b = [b_1]$ et $a = [a_1]$, où :

$$a_1 = \begin{cases} +1 & \text{si } w_{1,1}\, p_1 + w_{1,2}\, p_2 \geq b_1 \\ -1 & \text{autrement} \end{cases}$$

Ainsi le réseau peut prendre seulement deux valeurs distinctes selon le niveau d'activation du neurone : –1 lorsque ce dernier est strictement inférieur à b_1 ; +1 dans le cas contraire. Cette condition délimite donc l'espace des entrées en deux régions correspondantes. Cette frontière est définie par la condition :

$w_{1,1}\, p_1 + w_{1,2}\, p_2 = b_1$

qui correspond à l'expression générale d'une droite dans le plan comme illustrée à la figure 2-8.

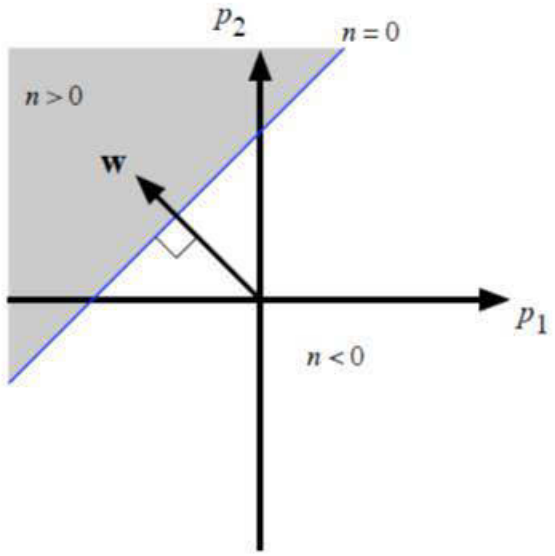


Figure 2- 8: Frontière de décision pour un perceptron monocouche à 1 neurone et 2 entrées

Dans ce type de problème, la sortie divise l'espace des entrées en deux régions : on dit que le problème est linéairement séparable.

Il a été prouvé que les perceptrons monocouche sont limités aux seuls problèmes linéairement séparables. Cette limitation a été levée avec la mise au point des réseaux multicouches. On a démontré par le passé qu'avec un perceptron de trois couches, deux couches cachées + une couche de sortie, comme celui de la figure 2-6, on peut construire des frontières de décision de complexité quelconque, ouvertes ou fermées, concaves ou convexes, à condition d'employer une fonction de transfert non linéaire et de disposer de suffisamment de neurones sur les couches cachées [28].

2.1.5 Les perceptrons multicouches

Un perceptron multicouche ou Multilayer Perceptron (MLP) en Anglais n'est rien d'autre qu'un assemblage de couches concaténées les unes aux autres, de la gauche vers la droite, en prenant les sorties d'une couche et en les injectant comme les entrées de la couche suivante.

Le problème de la navigation peut être vue de différentes manières:

- comme un problème à deux classes non linéairement séparables de type OU exclusif: passer ou ne pas passer, accélérer ou ne passer accélérer, tourner à droite ou tourner à gauche etc.
- comme un problème d'approximation de fonction. On suppose qu'il y a une loi qui régit toutes les réactions du conducteur aux stimuli qu'il reçoit. Et on tente de trouver un estimateur de cette loi,
- comme un problème de classification: tourner à droite, tourner à gauche, continuer tout droit. Accélérer, décélérer, maintenir sa vitesse etc.

Dans ce qui va suivre, nous essayerons de montrer par des exemples qu'un perceptron multicouche permet de résoudre ces types de problèmes au moyen de ses couches supplémentaires.

Une chose que l'on peut déjà remarquer est qu'il ne sert à rien d'assembler plusieurs couches linéaire (Adaline) car la combinaison de plusieurs couches linéaires peut toujours se ramener à une seule couche linéaire équivalente. Pour être utile donc, un réseau multicouche doit toujours posséder des neurones avec des fonctions de transfert non-linéaires sur ses couches cachées. Sur sa couche de sortie, selon le type d'application, il pourra comporter des neurones linéaires ou non-linéaires.

a) Le problème du OU exclusif

La réalisation de la fonction OU exclusif est un cas de problème à deux classe non linéairement séparables que l'on résout au moyen d'un perceptron multicouche. Mais qu'entend-on par problème à deux classes «linéairement séparables» ? Simplement un problème de classification dont la frontière de décision permettant de séparer les deux classes peut s'exprimer sous la forme d'un hyperplan (plan dans un espace à n dimensions ; dans un espace à deux dimensions, la frontière est une droite). Soit par exemple les problèmes illustrés sur la figure 2-9 où on veut séparer les points noirs des points blancs. Aucun de ces problèmes n'est séparable au moyen d'une droite). Des frontières possibles sont dessinées en pointillés. Elles sont toutes non linéaires.

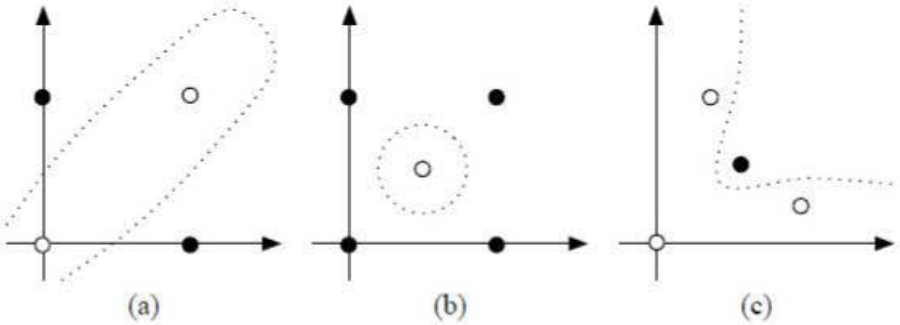


Figure 2- 9: Exemples de problèmes non linéairement séparables

C'est le problème du «OU exclusif» (XOR) que l'on ne peut pas résoudre ni avec un perceptron simple, ni avec un réseau Adaline, car les points noirs ne peuvent pas être séparés des blancs à l'aide d'une seule frontière de décision linéaire. Dans ce problème, les points noirs représentent le vrai (valeur 1) et les points blancs le faux (valeur 0). Le «ou exclusif», pour être vrai, exige qu'une seule de ses entrées soit vraie, sinon il est faux. On peut résoudre facilement ce problème à l'aide du réseau multicouche illustré à la figure 2-10 [28], fait de deux couches avec des fonctions de transfert seuil et une combinaison judicieux de poids et de biais. Sur la première couche, chaque neurone engendre les frontières de décision illustrées aux figures 2-1 a) et 2-11 b). Les zones grises représentent la région de l'espace d'entrée du réseau pour laquelle le neurone correspondant produit une réponse vrai. Le rôle du neurone de la couche de sortie, illustré à la figure 2-11 c), consiste à effectuer la conjonction des deux régions produites par les neurones de la première couche. La figure 2-11 représente toutes les frontières de décision dans l'espace des entrées. La frontière de décision engendrée par le neurone de la couche de sortie est aussi illustrée dans son propre espace d'entrée à la figure 2-12.

Mentionnons finalement que le réseau de la figure 2-10 n'est pas le seul à résoudre le problème du «ou exclusif». D'autres combinaisons de poids et de biais pourraient produire le même résultat.

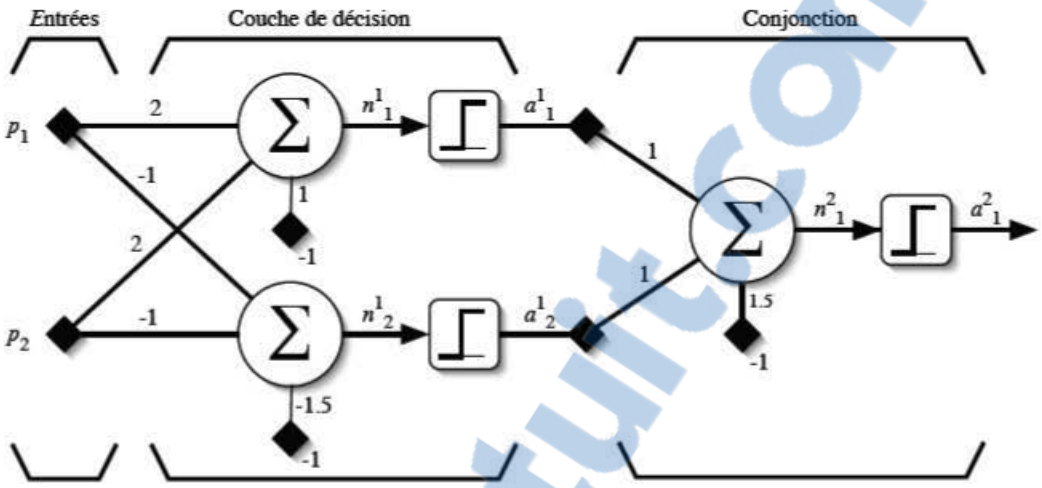


Figure 2- 10: Un perceptron pour le problème du OU exclusif

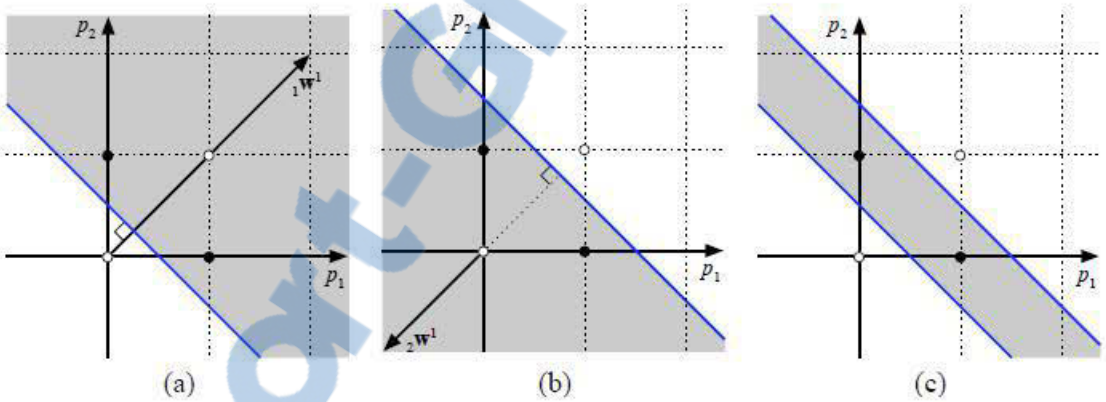


Figure 2- 11: Frontières de décisions engendrées par le réseau de la figure 2-15 :
(a) neurone 1 de la couche 1 ; (b) neurone 2 de la couche 1 ; (c) neurone 1 de la couche 2.

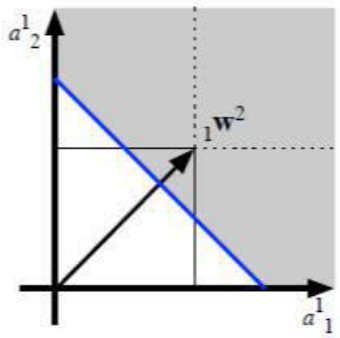


Figure 2- 12: Frontière de décision engendrée par le neurone qui effectue une conjonction

b) L'approximation de fonction

Pour faire de l'approximation de fonction, on peut montrer qu'un réseau multicouche comme celui de la figure 2-13, avec une seule couche cachée de neurones sigmoïdes et une couche de sortie de neurones linéaires permet d'approximer n'importe quelle fonction avec une précision arbitraire, à condition de disposer de suffisamment de neurones sur la couche cachée [28]. Intuitivement, c'est comme pour les séries de Fourier qui utilisent des sinus et cosinus. N'importe quelle fonction peut être écrite sous forme de combinaison linéaire de sigmoïdes.

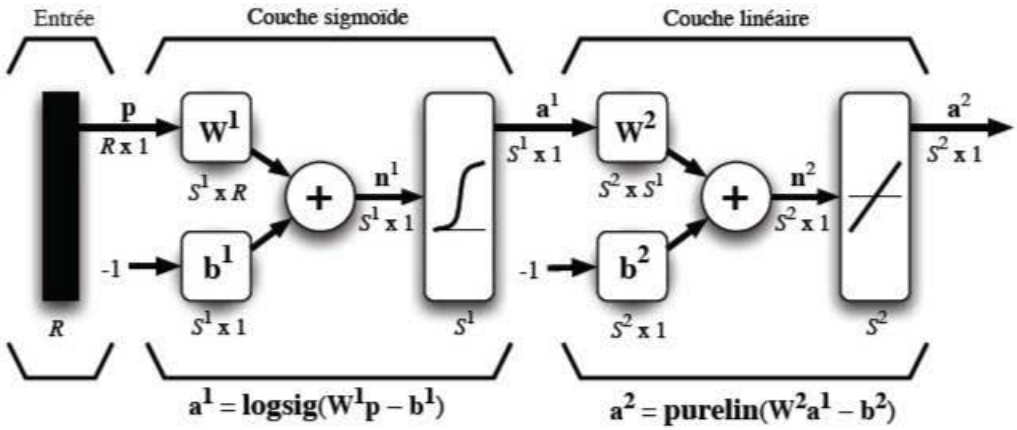


Figure 2- 13: Réseau multicouche permettant de faire de l'approximation de fonction

c) La classification

Pour faire de la classification, on utilise des réseaux soit à deux, soit à trois couches (comme à la figure 2-6) de neurones sigmoïdes. On peut montrer qu'une seule couche cachée suffit à engendrer des frontières de décision convexes, ouvertes ou fermées, de complexité arbitraire, alors que deux couches cachées permettent de créer des frontières de décision concaves ou convexes, ouvertes ou fermées, de complexité arbitraire. La figure 2-14 montre en deux dimensions différents types de frontières de décision [28]. Intuitivement, on peut imaginer que la première couche cachée d'un tel réseau sert à découper l'espace d'entrée en plusieurs régions à l'aide de frontières de décision linéaires ou non linéaires. La deuxième couche sert à faire la conjonction des régions délimitées par la couche précédente et il en résulte des frontières de décision non-linéaires convexes. De même, la couche de sortie fait la conjonction des régions délimitées par la couche précédente et il en résulte des frontières de décision concaves.

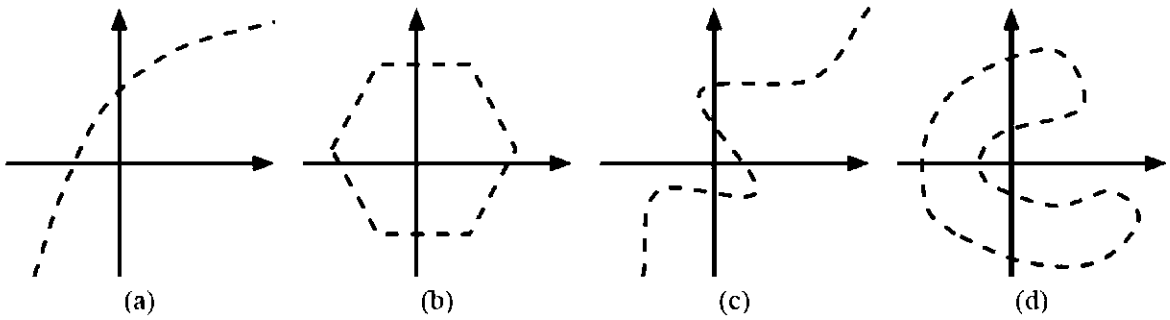


Figure 2- 14: Exemples de frontières de décision : (a) convexe ouverte ; (b) convexe fermée; (c)concave ouverte ; et (d) concave fermée

2.2 L'apprentissage des réseaux de neurones

Le champ des réseaux de neurones a pour objet de répondre à la question suivante : comment pouvons nous construire un système informatique qui automatiquement améliore ses performances avec l'expérience et quel est le principe organisateur du processus d'apprentissage ? Il s'agit, très schématiquement, de concevoir des algorithmes capables de construire des schémas généraux des choses à partir de cas particuliers et de se servir de ces schémas pour inférer sur d'autres cas particuliers. L'apprentissage est un processus de généralisation au cours duquel à partir d'exemples, on construit un modèle qui nous permettra de reconnaître de nouveaux exemples. En fait la caractéristique finale de

l'apprentissage est de nous faire bénéficier d'un retour d'expérience chaque fois que nous avons à refaire une chose que nous avons déjà faite au moins une fois dans le passé. Autrement dit, l'apprentissage permet quand vient le temps de refaire quelque chose qu'on a déjà faite de:

1) Mieux la refaire (diminution du nombre d'erreur)
2) La refaire rapidement (diminution du temps d'exécution)
3) La refaire plus facilement (diminution de la ressource mentale et/ou physique investie)

Cette définition implique qu'un réseau se doit d'être stimulé par l'environnement, qu'il subisse des changements en réaction à cette stimulation, et que ceux-ci provoquent dans le futur une réponse nouvelle vis-à-vis de l'environnement. Ainsi, le réseau peut s'améliorer avec le temps.

Dans la plupart des architectures, l'apprentissage se traduit par une modification de l'efficacité synaptique, c'est-à-dire par un changement de la valeur des poids qui relient les neurones d'une couche à l'autre.

On distingue deux grandes catégories de réseaux de neurones : les réseaux feed-forward (ou réseaux non-bouclés) et les réseaux feed-back (ou réseaux bouclés ou réseaux récurrents). Parmi les principales architectures non-bouclées, on compte le perceptron, le perceptron multicouche, les réseaux à fonction radiales (RBF). Elles servent à la classification et l'approximation de fonction. Et dans la catégorie des réseaux bouclés, il y a les réseaux à compétition dont les cartes auto-organisatrices de Kohonen, les réseaux de Hopfield et les réseaux ART [28].

Les réseaux non-bouclés sont entrainés essentiellement au moyen du mode d'apprentissage dit supervisé pendant que les réseaux bouclés utilisent l'apprentissage par renforcement et l'apprentissage non-supervisé.

2.2.1 L'apprentissage supervisé

C'est l'apprentissage par des exemples que donne un «professeur» à un «élève » durant une leçon ou un cours. C'est le mode d'apprentissage le plus utilisé dans les applications de réseau de neurones. En pratique, les exemples sont constitués d'une quantité Q de couples associant une sortie à une entrée $\{(p_1, d_1), (p_2, d_2), \ldots, (p_Q, d_Q)\}$, où p_i désigne un stimulus (entrée) et d_i la sortie désirée ou ortie cible pour ce stimulus. L'élève est représenté par le réseau de neurones à instruire et le professeur par l'ensemble des exemples.

Conceptuellement, le principe est illustré par le schéma suivant [28]:

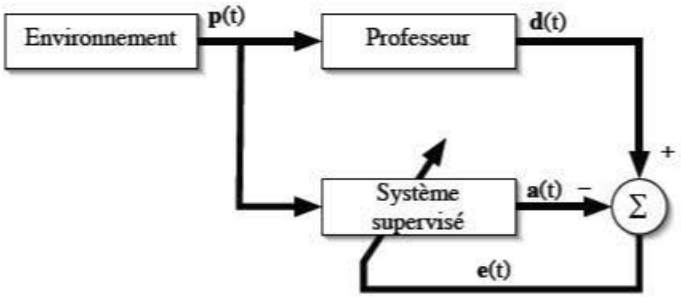


Figure 2- 15: Schéma bloc de l'apprentissage supervisé

L'environnement est inconnu du réseau. Celui-ci produit un stimulus p(t) qui est acheminé à la fois au professeur et au réseau. Grâce à ses connaissances intrinsèques, le professeur produit une sortie désirée d(t) pour ce stimulus. On suppose que cette réponse est optimale. Elle est ensuite comparée (par soustraction) à la sortie a(t) du réseau pour produire un signal d'erreur e(t) qui est réinjecté dans le réseau afin de modifier son comportement via une procédure itérative qui, éventuellement, lui permet d'émuler la réponse du professeur. Autrement dit, la connaissance de l'environnement par le professeur est graduellement transférée au réseau jusqu'à l'atteinte d'un certain critère de satisfaction qui constitue le critère d'arrêt.

Concrètement, modifier le comportement d'un réseau de neurones consiste à modifier ses poids comme définis au paragraphe 2.1.3 consacré au modèle d'un réseau de neurones. On peut considérer les instances des entrées comme des "questions" et les valeurs cible comme des "réponses". Ainsi, à chaque fois qu'un couple d'entrée-cible est présenté au réseau de neurones, ce dernier connaît la réponse pour une question donnée. Il n'en demeure pas moins qu'à chaque instance, le réseau de neurones doit faire une prévision à l'aide des valeurs actuelles des poids, et ses performances sont alors testées en évaluant la différence entre sa prévision et la valeur désirée qui constitue donc l'erreur de prévision. Les erreurs de toutes les instances sont accumulées et leur moyenne quadratique est calculée. Si cette moyenne n'est pas adéquate c'est-à-dire inférieure à un certain seuil choisi, les poids du réseau sont ajustés afin de produire une réponse plus exacte au prochain cycle d'apprentissage.

D'une manière générale, ce processus d'apprentissage intègre un certain bruitage (c'est-à-dire que les réponses du réseau peuvent être parfois plus exactes lors du cycle précédent de l'apprentissage par rapport au cycle actuel), mais en moyenne, l'importance des erreurs s'amenuise à mesure que l'apprentissage du réseau progresse. C'est pourquoi, on considère la moyenne des erreurs plutôt que les erreurs individuelles. L'ajustement des poids s'effectue par le biais d'un algorithme d'apprentissage, qui comme un professeur,

va apprendre au réseau de neurones à choisir ses poids afin d'obtenir de meilleures prévisions pour chacun des couples d'exemples entrée-cible du fichier de données appelé base d'apprentissage.

Du point de vue des calculs, l'apprentissage suit les étapes suivantes:

1) Présenter au réseau un couple entrée-sortie désirée.
2) Calculer la prévision du réseau en fonction des poids actuels.
3) Calculer la différence entre la prévision du réseau et la sortie désirée.
4) Reprendre les étapes 1 à 3 jusqu'à ce que tous les couples entrée-sortie aient été présentés au réseau.
5) Calculer la moyenne quadratique des erreurs.
6) Si la moyenne quadratique est inférieure à la moyenne précédente et supérieure au seuil, ajuster les poids et retourner à 1).
7) Sinon s'arrêter et donner la matrice des poids et la moyenne quadratique.

Un cycle d'apprentissage est appelé époque. Comme le montre l'algorithme, le nombre d'époques nécessaire pour entraîner un réseau de neurones n'est pas connu a priori mais défini dans le cadre du processus d'apprentissage. Un processus d'apprentissage type peut exigé plusieurs centaines d'époques.

2.2.2 L'apprentissage par renforcement

L'apprentissage par renforcement permet de contourner certaines des limitations de l'apprentissage supervisé. Au lieu d'un signal d'erreur vectoriel, il utilise un indice de satisfaction scalaire comme critère de contrôle de la réponse du réseau aux stimili. Ce type d'apprentissage est inspiré d'un énoncé de psychologie expérimentale de Thorndike datant de 1911qui dans le contexte des réseaux de neurones artificiels peut se résumer comme suit [28]:

Lorsqu'une action (décision) prise par le réseau engendre un indice de satisfaction positif, la tendance du réseau à prendre cette action doit être renforcée.

En pratique, l'usage de l'apprentissage par renforcement est complexe à mettre en œuvre. L'apprentissage supervisé dispose d'un signal d'erreur qui non seulement permet de calculer un indice de satisfaction (p.ex. l'erreur quadratique moyenne), mais permet aussi d'estimer le gradient local qui indique une direction pour l'adaptation des poids synaptiques.

Dans l'apprentissage par renforcement, l'absence de signal d'erreur rend le calcul de ce gradient impossible. Pour estimer le gradient, le réseau est obligé de tenter des actions et d'observer le résultat, pour éventuellement inférer une direction de changement pour les poids synaptiques. Pour ce faire, on implante un processus d'essais et d'erreurs tout en retardant la récompense offerte par l'indice de

satisfaction. Ainsi, on introduit deux étapes distinctes : une d'exploration où l'on essaie des directions aléatoires de changement, et une d'exploitation où l'on prend une décision. Ce processus en deux étapes peut ralentir considérablement l'apprentissage. De plus, il introduit un dilemme entre le désir d'utiliser l'information déjà apprise dont on est sûr et celui d'acquérir de nouvelles connaissances.

2.2.3 L'apprentissage non-supervisé

Elle est caractérisée par l'absence complète de professeur, c'est-à-dire qu'on ne dispose ni d'un signal d'erreur, comme dans le cas supervisé, ni d'un indice de satisfaction, comme dans le cas par renforcement. Nous ne disposons donc que d'un environnement qui fournit des stimuli, et d'un réseau qui doit apprendre sans intervention externe. Cet apprentissage est souvent synonyme de clustering (segmentation, regroupement) : regroupement par catégories des exemples entrants. En assimilant les stimuli de l'environnement à une description de son état interne, la tâche du réseau est de modéliser cet état le mieux possible. Pour y arriver, il importe d'abord de définir une mesure de la qualité pour ce modèle, et de s'en servir par la suite pour optimiser les paramètres libres du réseau, c'est-à-dire ses poids synaptiques. A la fin de l'apprentissage, le réseau a développé une habilité à former des représentations internes des stimuli de l'environnement permettant d'encoder les caractéristiques de ceux-ci et, par conséquent, de créer automatiquement des classes de stimuli similaires.

L'apprentissage non-supervisé s'appuie généralement sur un processus compétitif permettant d'engendrer un modèle où les poids synaptiques des neurones représentent des prototypes de stimuli. La qualité du modèle résultant doit s'évaluer à l'aide d'une métrique permettant de mesurer la distance entre les stimuli et leurs prototypes. C'est le processus de compétition qui permet de sélectionner le prototype associé à chaque stimulus en recherchant le neurone dont le vecteur de poids synaptique est le plus proche (au sens de la métrique choisie) du stimulus en question.

2.2.4 Revue de littérature des approches neuronales de la navigation autonome

Certainement les outils de modélisation non-linéaires les plus utilisés, les réseaux de neurones peuvent en théorie comme nous l'avons précédemment évoqué approximer n'importe quelle fonction continue en découvrant leur structure durant une phase d'apprentissage. Il existe beaucoup de publications en robotique mobile où les réseaux de neurones sont utilisés seuls ou en combinaison avec la logique floue ou d'autres outils pour la planification de mouvement. Toutes ces études procèdent de la même façon: construire une stratégie de navigation réactive et l'implémenter au moyen d'un réseau de neurones. On

rencontre la plupart des architectures: perceptrons multicouches, cartes auto-organisatrices de Kohonen, réseaux de Hopfield et même les réseaux ART.

Dans leurs travaux, S. Yang et M. Meng [39] ont proposé un modèle de réseau de neurones capable de guider un robot dans un environnement réel qui utilise l'activité dynamique du réseau, sans recherche explicite d'espace libre ou chemins sans collision, sans optimisation explicite de fonction de coût , sans aucune connaissance préalable de la dynamique de l'environnement , sans aucun processus d'apprentissage, et sans aucune procédure d'évitement de collision.

Un neuro-contrôleur intelligent pour la navigation d'un robot mobile a été présenté par Singh et Parhi [35]. Dans cette application, on donne la distance de l'obstacle de droite, la distance de l'obstacle de gauche, la distance de l'obstacle du centre et la direction de la cible comme entrées au réseau de neurones qui donne comme sortie l'angle de braquage du robot.

Un survol des types, architectures , algorithmes de base et problèmes pouvant être résolu à l'aide des réseaux de neurones est présentée dans les travaux de Tripathi et Rihani [37] . Les applications des réseaux de neurones pour les problèmes de classification et de reconnaissance des formes sont bien connues . Des solutions intéressantes aux problèmes de classification de type zones occupées/zones libres ont été obtenues dans le domaine de la navigation robotique au moyen de réseau de neurones de type compétitif. Les réseaux de neurones compétitifs sont aussi utilisés pour le contrôle et la génération de trajectoire de robot de même que pour le traitement de données capteur pour la mise à jour de carte de navigation [15] . On retrouve aussi l'utilisation de réseaux neurones récurrents entrainés avec l'algorithme de rétro-propagation du gradient dans des applications de planification de mouvement avec évitement d'obstacles.

T. Zhao, Y. Wang [41] ont utilisé un perceptron multicouche pour planifier les mouvements d'un robot. Dans cette application, on a voulu qu'un robot sorte de lui-même d'un labyrinthe. Ici trois sonars sont disposés à gauche, au centre et à droite du robot qui recueillent respectivement la distance de l'obstacle le plus proche à gauche, la distance de l'obstacle le proche au centre et la distance de l'obstacle le plus proche à droite. Les trois distances sont transmises en entrée au réseau de neurones. La direction à suivre est simplement celle du sonar ayant retourné la distance la plus longue.

Dans [37], Tripathi et Rihani utilisent aussi un perceptron multicouche pour le guidage d'un robot mobile. Dans cette application, on a considéré un capteur infrarouge pour la mesure des distances entre le robot et les obstacles qui balaie le devant du robot entre -90° et 90°. Cinq directions sont sondées: -90°, -45°, 0°, 45°, 90° et les distances mesurées sont les entrées du neuro-contrôleur qui décide de diriger le robot vers

la droite, la gauche ou tout droit et comme dans le cas précédent la décision consiste à prendre la direction de la distance la plus longue. Il a été remarqué que ce perceptron multicouche réussit à sortir convenablement le robot d'un labyrinthe.

2.3 Avantages et inconvénients des réseaux de neurones

Les réseaux neuronaux sont des classificateurs naturels ayant d'intéressantes caractéristiques comprenant mais sans être limitées à:

1) La capacité de représenter n'importe quelle fonction linéaire ou pas, complexe ou simple
2) La faculté d'apprendre à construire le modèle sous-jacents au processus en exploitant des exemples représentatifs de son fonctionnement.
3) La robustesse aux bruits ou au manque de fiabilité des données
4) Le traitement parallèle

Au nombre des inconvénients il y a:

1) L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans la (ou les) couche(s) cachée(s).
2) Le problème du sur-apprentissage ou apprentissage par cœur, qui nuit à la généralisation

Les inconvénients ne sont pas nombreux, mais ils sont très difficiles à contrer particulièrement pour ce qui a trait au contrôle de la qualité de l'apprentissage. Le problème est de juger de la pertinence de la base d'apprentissage c'est-à-dire de dire jusqu'à quel point les exemples présentés au réseau de neurones sont représentatifs du problème. Ce problème de représentativité concerne les exemples pris, individuellement comme collectivement. Deux exemples peuvent être individuellement représentatifs et constituer une base d'apprentissage qui ne l'est pas à cause des liens qu'ils entretiennent entre eux comme des liens de redondance par exemple. Nous reviendrons plus en détail sur ce problème au chapitre 4 après avoir présenté la stratégie de navigation que nous proposons au chapitre 3.

CHAPITRE 3

GUIDAGE D'UN ROBOT MOBILE PAR UN PERCEPTRON MULTICOUCHE

Dans ce chapitre, nous exposons notre approche pour la navigation autonome. Nous décrivons l'architecture de commande que nous proposons et ses composantes: la stratégie de perception de l'environnement, la stratégie pour le guidage latéral et celle pour le guidage longitudinal.

3.1 L'architecture de commande

Soit RO le robot se déplaçant dans l'espace W représenté comme un sous-espace d'un espace vectoriel de dimension 2, et dans lequel sont aussi présents des obstacles.

Le problème se pose comme ceci: d'une position de départ Pa à une position destination Pz on veut que le robot se déplace dans W sans jamais entrer en collision avec un obstacle soit en changeant de direction soit en réduisant sa vitesse.

La réalisation de cet processus nécessite un module de perception, un module de planification et un module de guidage qui travaille en coopération pour délivrer aux contrôleurs du véhicule des consignes de direction et de vitesse appropriées.

Le système de commande neuronal doit assurer la fonction de guidage. Ceci consiste à calculer à partir des points de passage dictés par le module de planification (mode autonome) ou par l'opérateur (mode télé-planifié), les consignes de cap et de vitesse à délivrer aux contrôleurs de direction et de vitesse du véhicule. Le module de planification définit les points de passage indépendamment de l'état actuel de l'environnement (présence d'obstacles).

Les techniques de commande neuronale que nous utilisons sont de type direct, c'est-à-dire qu'elles n'utilisent pas un modèle du véhicule. Il n'est donc pas nécessaire de réaliser préalablement la modélisation de la dynamique du véhicule pour les comportements latéral et longitudinal. Notre objectif est de réaliser l'apprentissage d'un réseau de neurones dont l'entrée est une "image" de la route, et la sortie la consigne de direction et de vitesse , à l'aide d'un "professeur" humain.

D'après des résultats biens connus dans le domaine de la commande de robots mobiles soumis à des contraintes non-holonomes, il est possible d'asservir la position du véhicule sur une route en contrôlant conjointement son cap et sa vitesse [15, 18, 19]. Alors, nous proposons ici deux guidages indépendants,

un guidage latéral pour l'asservissement du cap, et un guidage longitudinal pour l'asservissement de la vitesse qui travaillent conjointement pour maintenir le véhicule sur la route sans collision.

La chaine de commande complète se présente comme suit:

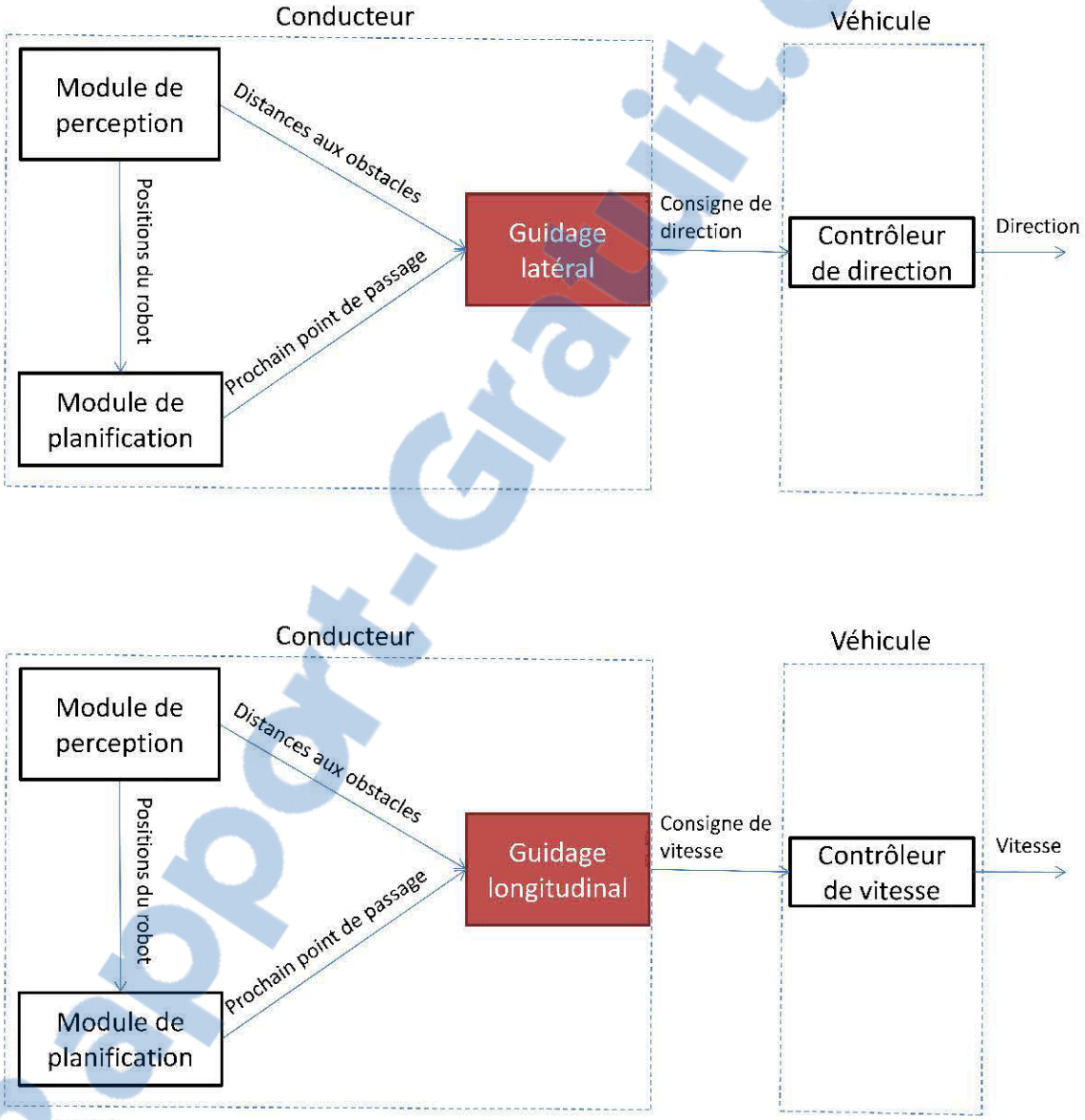


Figure 3- 1: Architecture de commande pour la navigation autonome

Sur la figure du haut qui représente la chaine de commande de direction, le module de guidage latéral exploite l'information sur la position du véhicule fourni par le module de perception et l'information sur le

prochain point de passage fourni par le module de planification pour donner une consigne de rotation au contrôleur de direction du véhicule.

Sur la figure du bas qui représente la chaine de commande de vitesse, le module de guidage longitudinal exploite l'information sur la position du véhicule fourni par le module de perception et l'information sur le prochain point de passage fourni par le module de planification pour donner une consigne de vitesse au contrôleur de vitesse du véhicule.

3.2 La perception de l'environnement

Le module de perception a la fonction de détecter les obstacles qui pourraient se trouver sur le chemin du robot. La perception se fait au moyen de télémètres ou de caméras ou de la combinaison des deux procédés. Quelle que soit la technologie utilisée pour effectuer la mesure télémétrique, le capteur retourne généralement deux informations. La première donne l'angle de gisement Θ_s qui correspond à la direction de perception de l'obstacle. La seconde information représente la distance d séparant le véhicule de l'obstacle perçu. Cette technique de mesure permet donc de positionner les objets présents dans la scène par rapport au robot. Dans notre algorithme, nous divisons l'environnement devant le robot en neuf secteurs ou segments égaux chacun surveillé par des rayons de télémètre. Si l'axe central vertical est à 0°, alors les secteurs sont localisés entre -90° et +90° couvrant chacun un angle de $\Theta_s = 180°/9 = 20°$ comme le montre la figure 3-2. Il peut s'agir d'une constellation de plusieurs télémètres ou d'un seul dont le faisceau couvre tout le devant du robot de -90° à +90°. De chaque secteur i, on retourne la distance d_i et la direction Θ_i de l'obstacle le plus proche. La figure 3-3 montre un éparpillement d'obstacles devant le robot où on peut voir plus d'un obstacle dans un même secteur. Mais seul l'obstacle le plus proche est considéré chaque fois (figure 3-4).

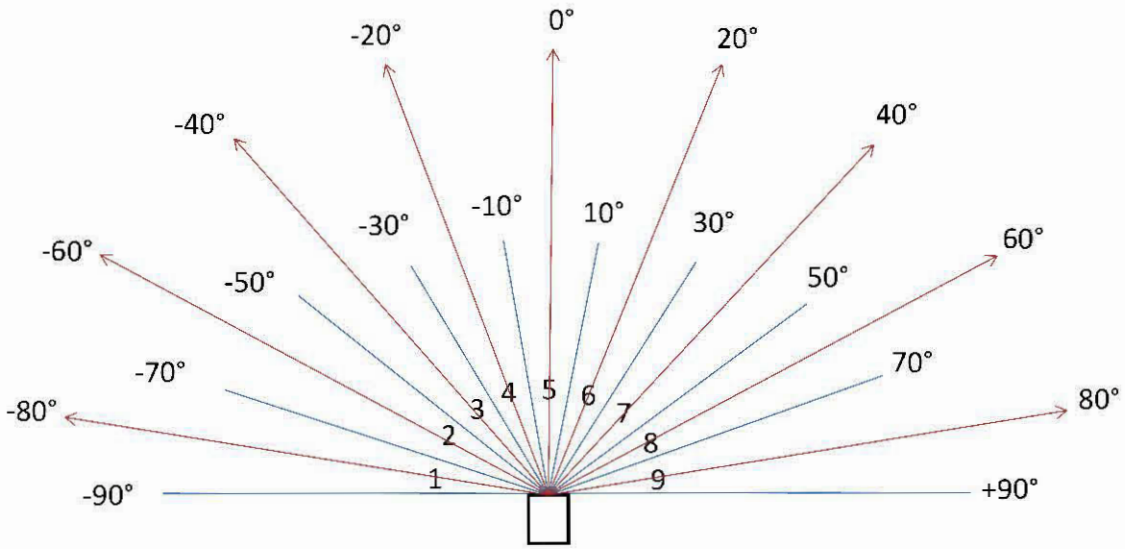


Figure 3- 2: Organisation du système de perception pour la détection d'obstacles

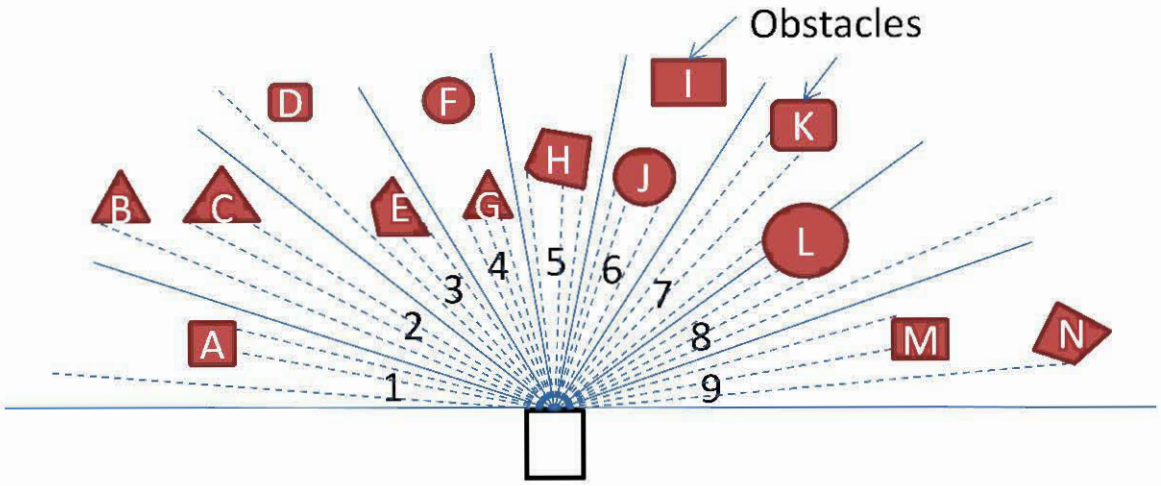


Figure 3- 3: Système perceptif pour la détection des obstacles: éparpillement d'obstacles devant le robot

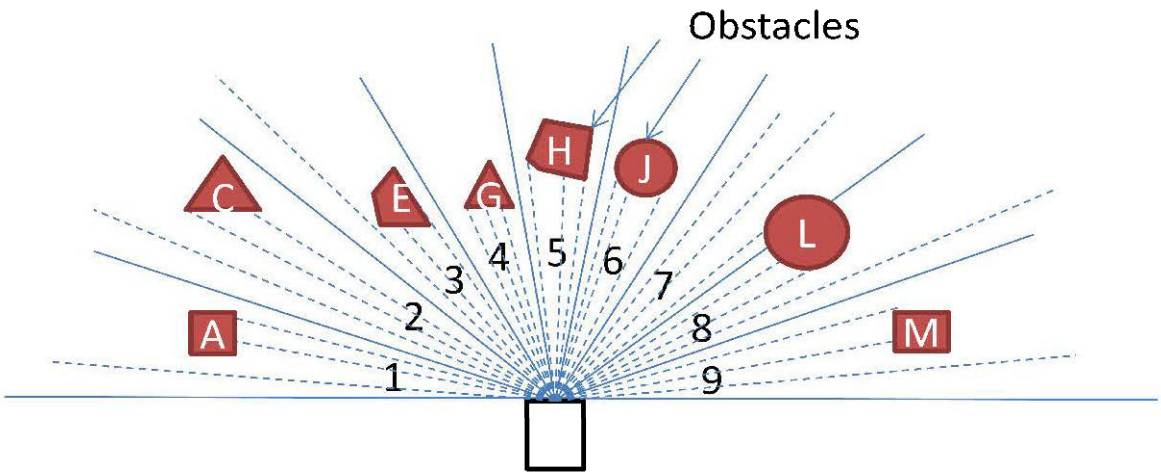


Figure 3- 4: Système perceptif pour la détection des obstacles: obstacles détectés

Le nombre de senseurs n'est pas limitatif, nous avons considéré neuf ici mais on peut en mettre autant que nécessaire en s'assurant d'une couverture uniforme de l'environnement.

3.3 La planification de la route

Se déplacer d'un endroit Pa à un endroit Pz, nécessite de connaitre la route menant de Pa à Pz c'est-à-dire l'ensemble des points de passage intermédiaires qui permettent de joindre Pa à Pz. Dans les intersections par exemple il est nécessaire de se faire dicter ou de se dicter dans quelle direction il faut tourner.

Un point de passage ou de cheminement, (waypoint en Anglais) ou point de virage désigne un point de la route à atteindre où doit avoir lieu un changement de cap. Il peut être matérialisé par des relèvements sur des amers particuliers, des balises radio, des relevés de position GPS.

La route ou chemin est la séquence de points de passage entre le point de départ et le point d'arrivée. Chaque fois que le point de passage cible est approché d'une certaine distance, le planificateur de route change automatiquement pour le prochain point de passage sur la liste.

A l'activation d'une route, le planificateur de route considère que le premier segment à naviguer va de Pa à P1 avec P1 le premier point de passage et Pa le point de départ. Le planificateur de route peut être un simple récepteur GPS.

Il est important de ne pas attendre d'atteindre le point de passage cible avant de changer pour le point de passage suivant. Ce changement doit s'effectuer suffisamment tôt donc à une distance Dp du point de passage en cours suffisamment longue pour permettre les virages en douceur et suffisamment tard pour suivre la géométrie de la route (voir figure 3.5)

Implicitement, on définit donc le prochain point de passage en fonction de la courbure de la route au point actuel. Dp est la distance qui devrait encore séparé le véhicule du point de passage en cours au moment où le point de passage suivant est annoncé. On peut remarquer que plus le véhicule va vite plus tôt doit s'effectuer le changement de point de passage; ce qui veut dire que plus longue devrait être la distance Dp. Inversement moins vite va le véhicule plus tard doit s'effectuer le changement de point de passage; donc moins longue devrait être la distance Dp. Ainsi, la distance Dp est une fonction de la vitesse: Dp = F(V). En fait, le choix d'un point de passage évoluant à une distance Dp fonction de la vitesse revient à considérer le point de passage comme un obstacle mobile en avant du véhicule duquel il faut en permanence garder une distance de sécurité identique à la distance de sécurité qu'un véhicule doit garder de son précédent pour éviter d'entrer en collision avec lui au cas où ce dernier freinerait brusquement. La distance de sécurité est la distance d'arrêt c'est-à-dire la distance conventionnelle théorique nécessaire à un véhicule pour s'arrêter compte tenu de sa vitesse. Cette distance est en réalité le cumul de la distance de freinage, distance conventionnelle nécessaire à un véhicule pour passer de sa vitesse initiale à la vitesse nulle, et de la distance de perception-réaction, distance parcourue par un véhicule à vitesse constante pendant le temps de perception-réaction du conducteur.

En effet lors d'un freinage d'urgence, le temps que met une voiture à s'arrêter se décompose en deux parties:

a) Le temps de réaction du conducteur (le temps nécessaire au conducteur pour prendre conscience de la situation, et appuyer sur le frein).
b) Le temps de freinage lui-même

Le temps de perception-réaction est constitué du temps physiologique de perception-réaction (1,3 à 1,5 s) et du temps mort mécanique d'entrée en action avec les freins (0,5 s).

Pour le calcul, on adopte généralement la valeur de 2 secondes pour le temps de perception-réaction quelle que soit la vitesse, même s'il est admis qu'en situation d'attention soutenue (vitesse supérieure à 100 km/h ou trafic soutenu à vitesse importante), ce temps peut être réduit à 1,8 s.

Toutefois, une modification de 0,2 s a très peu d'influence sur la distance d'arrêt et les différentes études tendent à situer cette valeur entre 2 et 2,5 s. Cette dernière valeur est d'ailleurs celle adoptée en Amérique du Nord.

Si V est la vitesse en m/s du véhicule et T_R le temps de réaction en seconde, la formule donnant la distance en mètres parcourue pendant le temps de réaction est:

$$D_R = \frac{V}{(3,6)^2} T_R = \frac{V}{12,96} T_R \quad (3.1)$$

La distance de freinage correspond à la distance parcourue par le véhicule pendant que sa vitesse diminue, c'est-à-dire à partir du moment où le conducteur appuie sur la pédale de frein.

Si V est la vitesse en m/s du véhicule, la distance de freinage en mètres peut être approximée à l'aide de la formule:

$$D_F = \frac{V^2}{2*g*\mu} = \frac{V^2}{19,62\mu} \quad (3.2)$$

avec g l'accélération de la pesanteur = 9.81m/s^2 et μ le coefficient d'adhérence qui se définit comme le rapport entre la décélération et une accélération de référence généralement prise égale à 10 m/s^2.

Le coefficient d'adhérence dépend de la nature et de l'état de la chaussée. Sur une route sèche, on a: μ = 0,8. Sur une route mouillée, on a: μ = 0,4. En moyenne, on considère μ =0,7

La distance d'arrêt (D_A) du véhicule est donc finalement la somme de ces deux distances :

$$D_A = D_R + D_F \quad (3.3)$$

Connaissant la distance de freinage, on peut calculer le temps de freinage. En effet, la décélération s'écrit:

$$\gamma = \frac{V^2}{2D_F} \quad (3.4)$$

avec V en m/s, D_F en m et γ en m/s^2.

Et le temps de freinage s'écrit:

$$T_F = \frac{V}{\gamma} = \frac{2D_F}{V} \quad (3.5)$$

Si nous considérons un temps de réaction de 2,5 secondes, alors la distance d'arrêt pour un véhicule se déplaçant à la vitesse V s'écrit:

$$D_A = 2,5V + V.T_F \quad (3.6)$$

$$D_A = 2,5V + 2D_F = 2,5V + 2\frac{V^2}{19,62\mu} = 2,5V + \frac{V^2}{9,81\mu}$$

La distance au point de passage Dp se calcule donc enfin:

$$Dp = D_A = 2{,}5V + \frac{V^2}{9{,}81\mu} \tag{3.7}$$

Dans la stratégie de navigation autonome que nous proposons dans ce travail, un module de planification de chemin doit dicter la route point de passage par point de passage aux modules de décision suivant le rythme évoqué précédemment. Cela peut se faire en lisant une carte de navigation pré-chargée ou générée en temps réel à partir des données d'un navigateur GPS. L'ordonnancement des points de passage est comme nous le verrons dans la suite, la fonction régulatrice du contrôle latéral (commande du cap) et du guidage longitudinal (commande de la vitesse) du robot.

Le module de guidage latéral dirige le robot vers le point de passage dicté si ce dernier est accessible et vers ailleurs sinon.

Le contrôleur longitudinal lui, répond constamment à la question de savoir quelle vitesse adopter pour garder une distance de l'obstacle le plus proche égale à la distance d'arrêt.

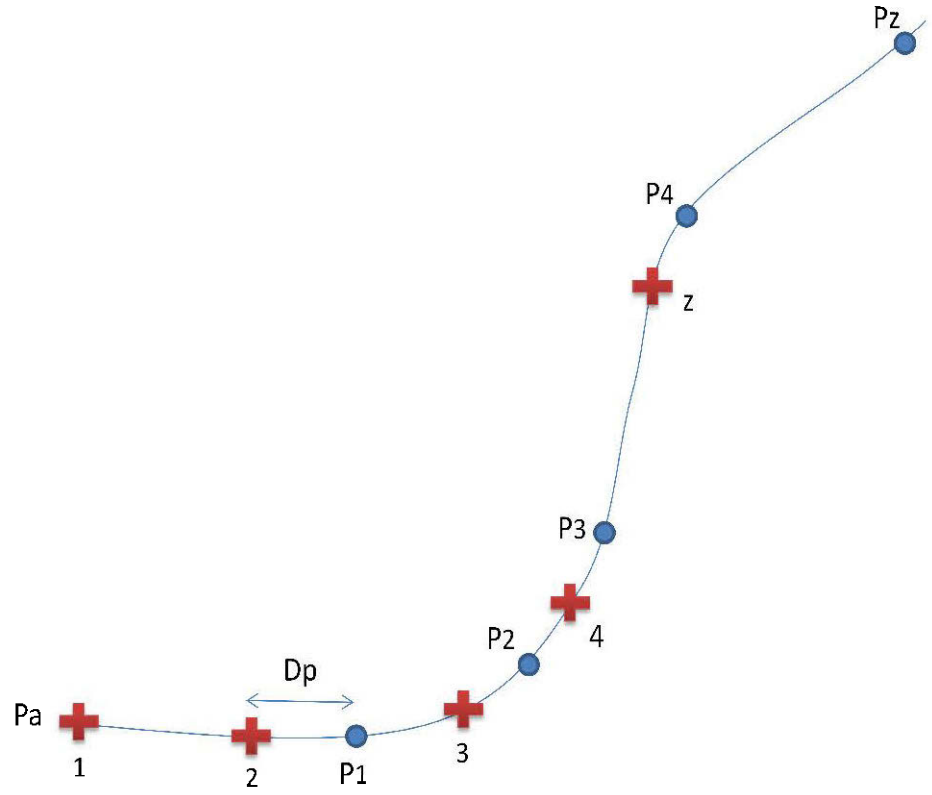


Figure 3- 5: Stratégie de lecture de la route

Sur la figure 3-5, P_1, P_2, P_3, P_4 et Pz sont les points de passage qui constituent la route et les croix 1, 2, 3, 4 et z sont les points où s'effectuent les changements de point de passage. Au point marqué par la croix 1, le point de cheminement cible est P_1. Au point marqué par la croix 2, on change pour P_2. Au point 3, on change pour P_3. Au point 4, on change pour P_4. Au point z on change pour P_z.

3.4 Le guidage latéral du mouvement: commande du cap

Le guidage latéral concerne la détermination des consignes de cap que le robot doit suivre durant son déplacement. A chaque itération, le module de guidage propose directement la consigne donnée par le planificateur de mouvement s'il n'y a pas d'obstacle à éviter. Sinon, il propose un cap alternatif permettant d'éviter l'obstacle.

3.4.1 L'algorithme pour le guidage latéral

Le robot peut prendre neuf directions possibles: -80°, -60°, -40°, -20°, 0°, +20°, +40°, +60°, +80° et les distances mesurées par les détecteurs d'obstacles sont comprises entre un minimum de 50 cm et un maximum de 250 cm. Les sondes ont forcément une portée maximale (ici 250 cm) et compte tenu de leurs positions sur le robot, elles ne peuvent détecter qu'à partir d'une certaine distance (ici 50 cm) (Figure 3-6)
.

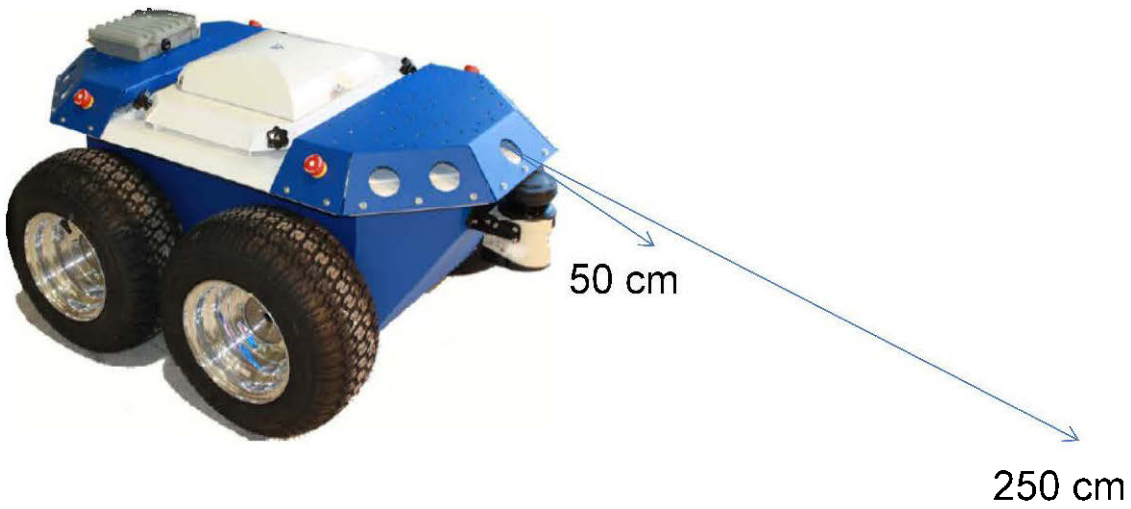


Figure 3- 6: Distances minimale et maximale mesurables

A chaque itération, les sondes retournent les distances aux obstacles dans les neufs secteurs et connaissant la position du prochain point de passage relativement au robot, le module de guidage latéral dicte la direction à prendre selon la stratégie décrite ci-dessous:

- *Prendre le segment où se trouve la cible (c'est-à-dire le point de passage en cours) si l'obstacle de ce segment est plus éloigné que la cible.*
- *Sinon prendre le segment dont l'obstacle est le plus éloigné.*

La figure 3-7 montre l'algorithme en diagramme tandis que la figure 3-8 présente deux situations pour expliquer son fonctionnement.

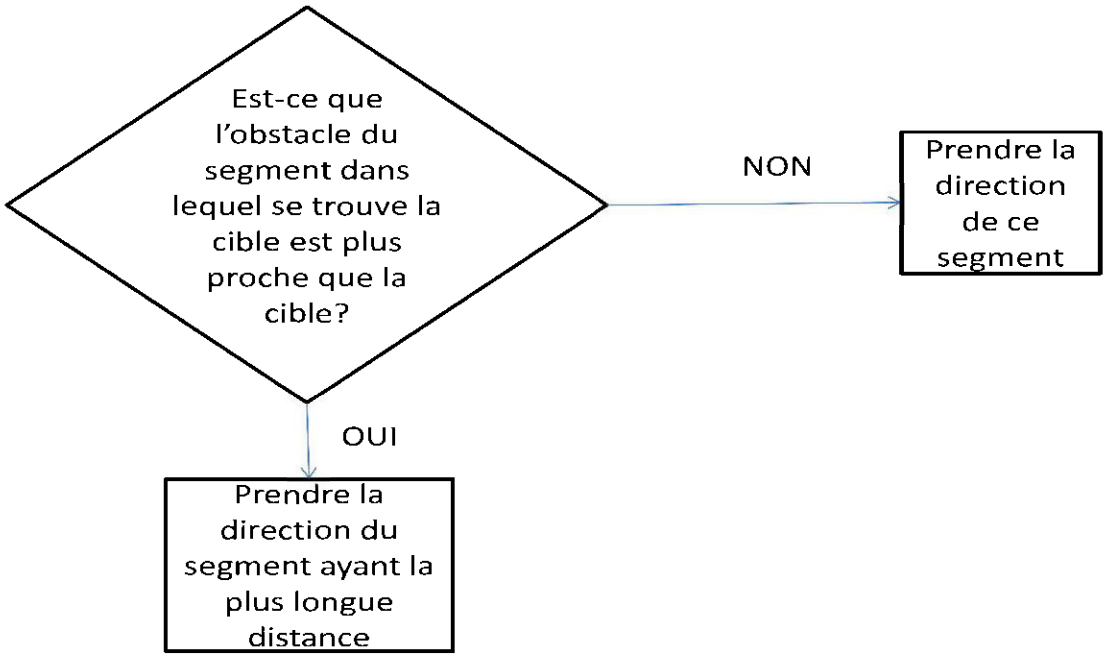


Figure 3- 7: Algorithme pour le contrôle du cap

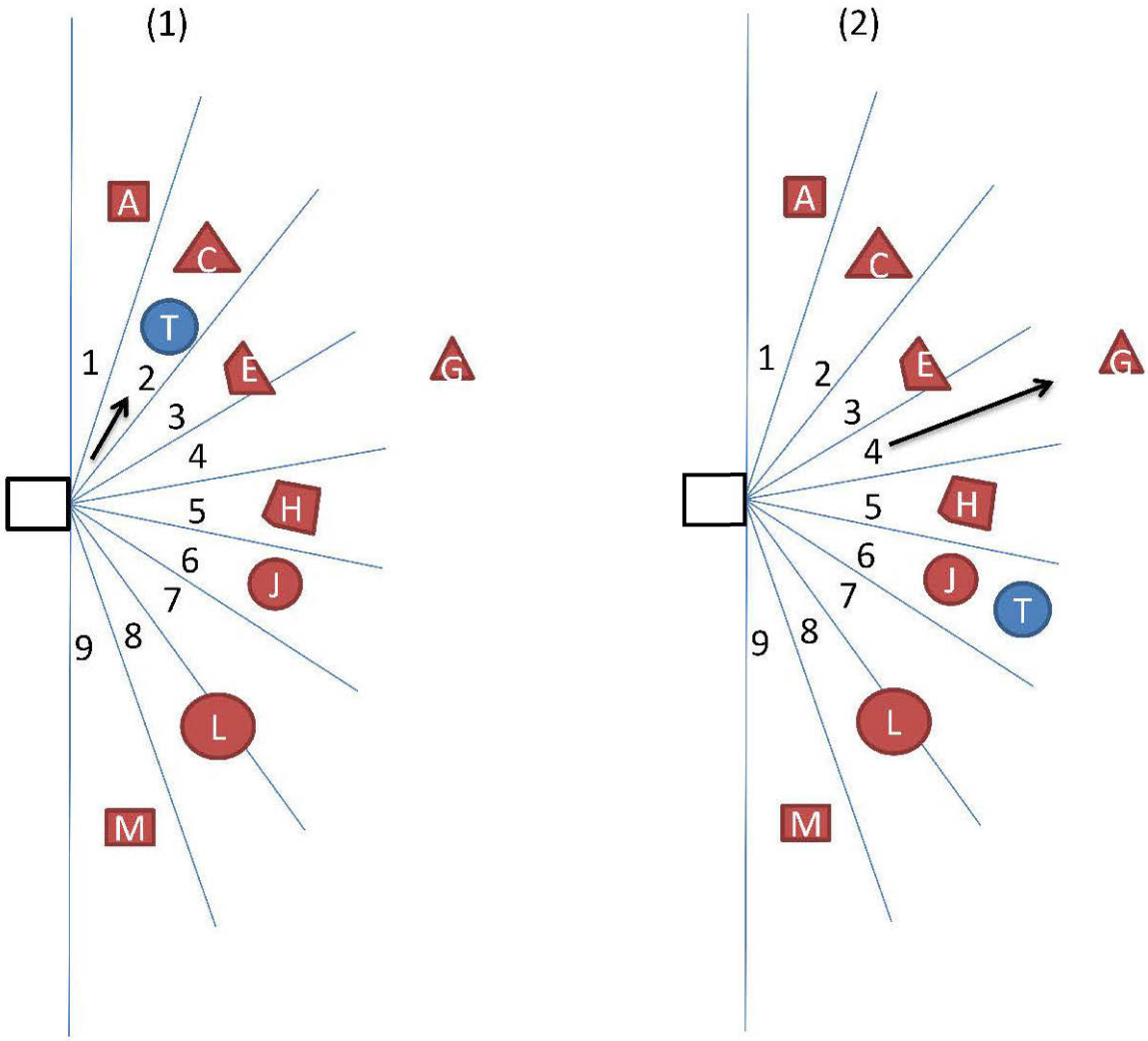


Figure 3- 8: Illustration de l'algorithme:

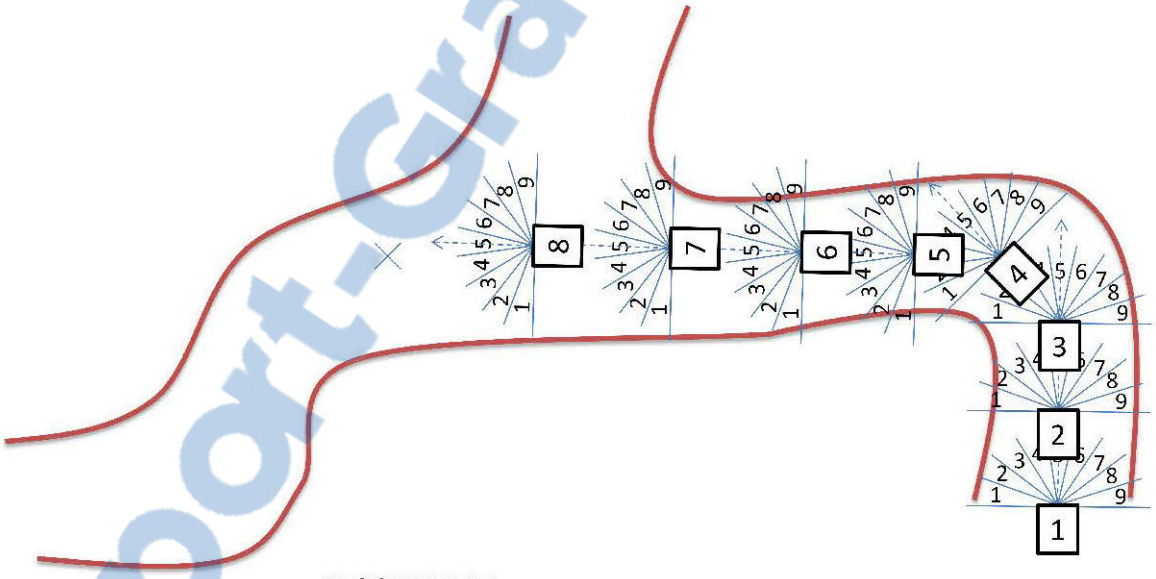
Les deux figures montrent deux configurations différentes d'obstacles. Sur la figure de gauche, la cible T est avant l'obstacle situé dans son segment. Par conséquent on peut prendre la direction de ce segment. Sur la figure de droite par contre, la cible T est après l'obstacle situé dans son segment. Par conséquent, ce segment doit être évité et on prend le segment ayant l'obstacle le plus éloigné c'est-à-dire le segment 4.

3.4.2 Validité de l'algorithme du guidage latéral

Il s'agit ici de montrer en quoi cet algorithme a les performances requises pour assurer la fonction de commande du cap dans le cadre de la navigation autonome. Pour cela, nous avons considéré deux itinéraires auxquels nous l'avons appliqué.

Dans l'exemple illustré par la figure 3.9 et le tableau 3.1, le robot suit une route exempte d'obstacle, sauf ceux que constituent ses bordures. Dans ce cas, comme le point-cible tient compte de la forme de la route, il doit être tout le temps suivi puisqu'il n'y a pas d'obstacle à éviter à l'intérieur des limites de la route.

Dans le deuxième exemple illustré par la figure 3.10 et le tableau 3.2, le robot est sur une route à l'intérieur de laquelle se trouvent des obstacles. Dans ce cas, l'évolution du robot est une succession de suivis de points-cibles et d'évitements d'obstacles.



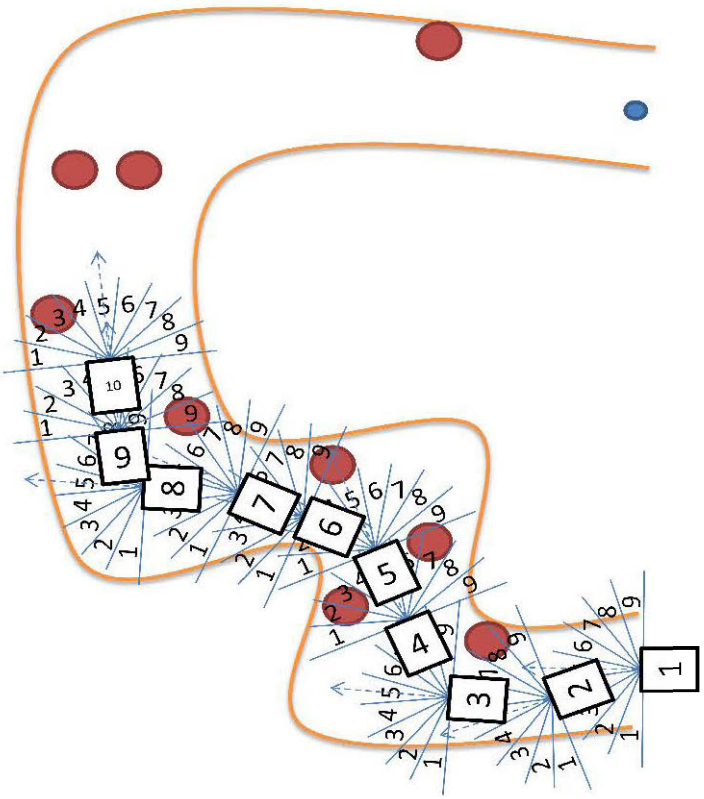
ALGORITHME
Prendre le segment où se trouve la cible si l'obstacle de ce segment est plus éloigné que la cible. Sinon prendre le segment dont l'obstacle est le plus éloigné.

Figure 3- 9: Exemple de suivi de route sans obstacle sur la route: ici seules les bordures de la route constituent des obstacles

Position	V1	V2	V3	V4	V5	V6	V7	V8	V9	T (Cible)
1					√					V5
2					√					V5
3			√							V3
4			√							V3
5					√					V5
6					√					V5
7					√					V5
8					√					V5

Tableau 3- 1: Illustration des positions successives suivies dans un cas de navigation sur route libre

Dans cet exemple, dans les positions 1, 2, 5, 6, 7 et 8 le segment cible est V5 et dans chacun des cas, le segment suivi est V5. Comme il n'y a jamais eu d'obstacle à éviter à l'intérieur de la route, les cibles dictées devaient être suivies.



ALGORITHME
Prendre le segment où se trouve la cible si l'obstacle de ce segment est plus éloigné que la cible. Sinon prendre le segment dont l'obstacle est le plus éloigné.

Figure 3- 10: Suivi d'une route avec des obstacles sur la route: ici en dehors des bordures de la route, il y a aussi des obstacles à l'intérieur de la route.

Position	V1	V2	V3	V4	V5	V6	V7	V8	V9	T (Cible)
1				√						V5
2						√				V6
3								√		V8
4					√					V5
5			√							V5
6					√					V5
7				√						V6
8									√	V9
9					√					V5
10					√					V5

Tableau 3- 2: Illustration des positions successives dans un cas de navigation sur route avec obstacles

Dans cet exemple-ci, la route est intérieurement jonchée d'obstacles. Dans la position 1, le segment cible est V5 mais on a suivi le segment V4 pour éviter un obstacle. Dans les positions 2, 3, 4, 6, 8, 9 et 10, les cibles sont respectivement V6, V8, V5, V5, V9, V5 et V5 et sont suivies. Dans les positions 1, 5 et 7, les cibles sont V5, V5 et V6 et les segments suivis ont été respectivement V4, V3 et V4 afin d'éviter des obstacles.

3.5 Le guidage longitudinal du mouvement: commande de la vitesse

Le guidage longitudinal concerne la détermination des consignes de vitesse que le robot doit adopter durant son déplacement. Le problème de la commande de vitesse ou guidage longitudinal vise à adapter la vitesse du véhicule aux distances aux obstacles pour empêcher les collisions avec les obstacles et aux angles de virage pour empêcher les sorties de route dans les virages.

Plus proche est le robot d'un obstacle, moins vite il doit se déplacer sous peine d'entrer en collision avec l'obstacle. Et plus loin il est de l'obstacle, plus vite il peut se déplacer sans risquer une collision.

Au sous-chapitre 3.3, nous avions présenté un algorithme de planification de la route où le prochain point de passage est défini en fonction de la courbure de la route au point où se trouve le mobile. En effet dans cet algorithme, on n'attend pas le point de passage cible avant de changer pour le prochain point de passage. Et ce changement effectue suffisamment en avance pour permettre les virages en douceur et suffisamment tard pour suivre la géométrie de la route. Cela fait que les points de passage se rapprochent dans les courbes et s'éloignent dans les lignes droites (voir figure 3.5). Dans le même algorithme, on traite

le point de passage comme un obstacle mobile en avant du véhicule duquel il faut en permanence garder une distance de sécurité.

Il apparait donc que l'ajustement de la vitesse aux angles de virage est équivalent à l'ajustement de la vitesse aux distances aux obstacles. Ainsi la règle de guidage longitudinal qui permet d'éviter une collision avec un obstacle est aussi celle qui permet d'éviter une sortie de route dans un virage. Nous ne croyons pas qu'il y a une formule pour la commande de vitesse pour l'évitement de collision avec les obstacles et une autre pour l'évitement de sortie de route dans les virages. Un seul et même algorithme s'applique dans les deux cas.

3.5.1 L'algorithme pour le guidage longitudinal

Notre stratégie ici est de déterminer la vitesse à adopter relativement aux obstacles frontaux c'est-à-dire ceux localisés dans les cinq segments du centre: 3, 4, 5, 6, 7 en utilisant la règle de la distance de sécurité. La considération des obstacles frontaux seuls se justifie par le fait que selon l'algorithme de commande latérale, les obstacles hors du champ visuel de ces 5 zones devraient toujours être esquivés par les manœuvres de changement de direction sans ajustement de vitesse.

Nous l'avons vu précédemment, la distance de sécurité ou encore distance d'arrêt, c'est la distance qu'un conducteur doit conserver entre son véhicule et celui qui le précède pour éviter d'entrer en collision avec lui en cas de freinage soudain de ce dernier. Lorsque deux véhicules se suivent en effet, le conducteur du second véhicule doit maintenir une distance de sécurité suffisante pour pouvoir éviter une collision en cas de ralentissement brusque ou d'arrêt subit du véhicule qui le précède. Cette distance est d'autant plus grande que la vitesse du véhicule suiveur est élevée. La formule (3.7) donne l'expression de cette distance en fonction de la vitesse. Elle tient compte comme nous l'avions dit des différents délais entre le calcul d'une vitesse par le module de guidage longitudinal, la transmission de cette vitesse au contrôleur de vitesse et son exécution par ce dernier.

A chaque itération, on considère la distance Dmin entre le robot et l'obstacle frontal le plus proche y compris le point de passage cible (considéré aussi comme un obstacle frontal): Dmin = Min(d3, d4, d5, d6, d7, Dp). La distance d'arrêt doit être plus courte que Dmin car on veut que le robot s'arrête avant d'avoir parcouru Dmin qui est ici la distance de collision.

Les distances mesurées par les détecteurs d'obstacles sont comprises entre un minimum de 50 cm et un maximum de 250 cm. Comme nous l'avons évoqué précédemment, les télémètres ont forcément une portée maximale (ici 250 cm) et compte tenu de leurs positions sur le robot, elles ne peuvent détecter

qu'au delà d'une certaine distance (ici 50 cm) (figure 3-7). Ainsi si on continue de s'approcher d'un obstacle alors qu'il est déjà à 50 cm, on va le perdre de vue et donc entrer en collision avec lui. Notre stratégie considère 50 cm comme la distance maximale d'approche. Autrement dit, la distance d'arrêt doit être telle que le véhicule s'arrête à 50 cm de l'obstacle. Cela peut être obtenu en considérant une distance d'arrêt D_A = Dmin - 0.5m.

Il s'agit ensuite de déterminer à quelle vitesse le véhicule doit amorcer le processus d'arrêt pour le compléter au bout d'une distance (Dmin - 0.5) mètres. Cette vitesse est la solution positive de l'équation:

$$Dmin - 0.5 = 2{,}5V + \frac{V^2}{9{,}81\mu}$$

On a donc:

$$V = \frac{-24{,}525\mu + 9{,}81\mu\sqrt{6{,}25 + \frac{4(Dmin - 0.5)}{9{,}81\mu}}}{2} \qquad (3.8)$$

avec V en m/s et Dmin en m.

Le tableau ci-dessous donne quelques valeurs de vitesses correspondantes à des distances D_A pour un coefficient d'adhérence de μ= 0,7

Distance Dmin (m)	0.5	0.55	0.65	0.9	1.02	1.2	1.45	1.65
Vitesse (m/s)	0	0.019977	0.059791755	0.158536	0.205539	0.275576	0.371942	0.448294
Vitesse (km/h)	0	0.071916	0.215250316	0.57073	0.739941	0.992075	1.33899	1.613857

Tableau 3- 3: Valeurs de vitesse pour des distances d'arrêt pour un coefficient d'adhérence de 0,7

3.5.2 Validité de l'algorithme du guidage longitudinal

La validité de cet algorithme va de soi. Dans une logique où on se prépare en permanence à s'arrêter à temps avant qu'une collision se produise, on peut éviter toutes les collisions prévisibles. Cependant l'efficacité d'une mesure d'arrêt d'urgence sera toujours limitée par la performance du système de freinage et les efforts que le véhicule pourra développer au niveau du contact pneumatique-chaussée.

3.6 Contrôle parallèle et conjoint du cap et de la vitesse

Ainsi comme le montre la figure 3-11, le contrôleur latéral a pour entrées les distances aux obstacles d1, d2, d3, d4, d5, d6, d7, d8, d9 et les coordonnées du prochain point de passage (ou cible): distance et direction Dp et θ. La sortie est l'orientation Λ. Les entrées du contrôleur longitudinal sont les distances aux obstacles frontaux d3, d4, d5, d6, d7 plus la distance du prochain point de passage: Dp. La sortie est la vitesse V.

Dans les deux cas, les distances d1, d2, d3, d4, d5, d6, d7, d8, d9 sont fournies par le système de perception et les coordonnées de la cible par le module de planification de route.

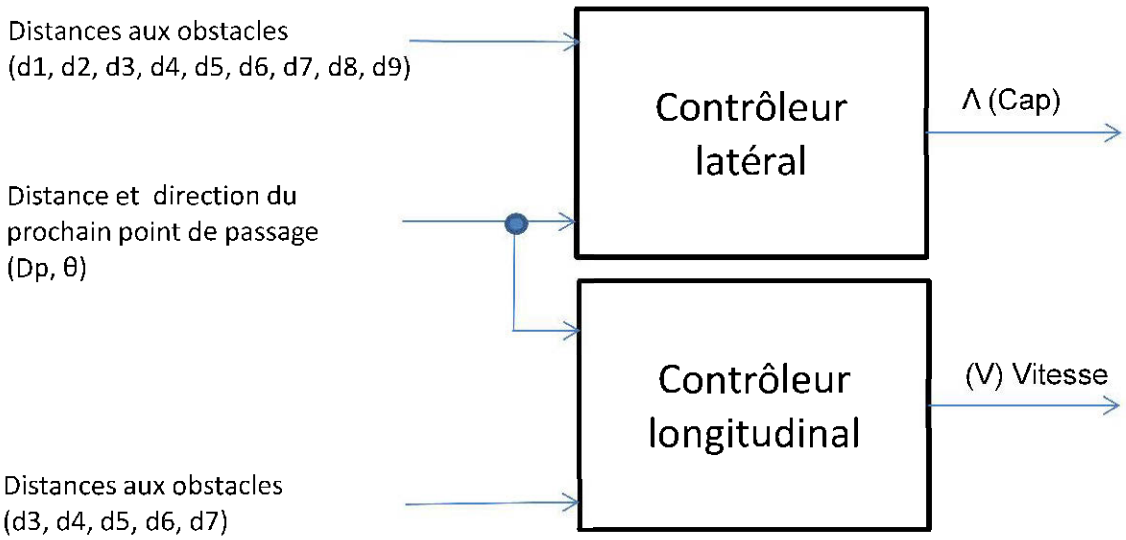


Figure 3- 11: Principe de la commande

3.7 Architectures des réseaux de neurones

Nous implémentons ici au moyen d'un perceptron multicouche les stratégies de commande du cap et de la vitesse précédemment proposées.

3.7.1 Architecture pour le contrôle latérale

La figure 3-12 montre le détail de l'architecture du contrôleur latéral. Nous avons comme entrées les neuf distances retournées par les détecteurs d'obstacles des segments: d1, d2, d3, d4, d5, d6, d7, d8, d9 auxquelles sont ajoutés les deux coordonnées de la cible (Dp et θ) ; il faut donc onze neurones d'entrées.

La sortie est le cap Λ à adopter; il y a donc un neurone sur la couche de sortie. Ce cap est transmis au module de pilotage du robot qui agit en conséquence sur ses actionneurs de direction pour réaliser ce cap. Pour le nombre de neurones cachés, nous avons essayé différents quantité pour trouver que douze neurones cachés donnent un résultat meilleur.

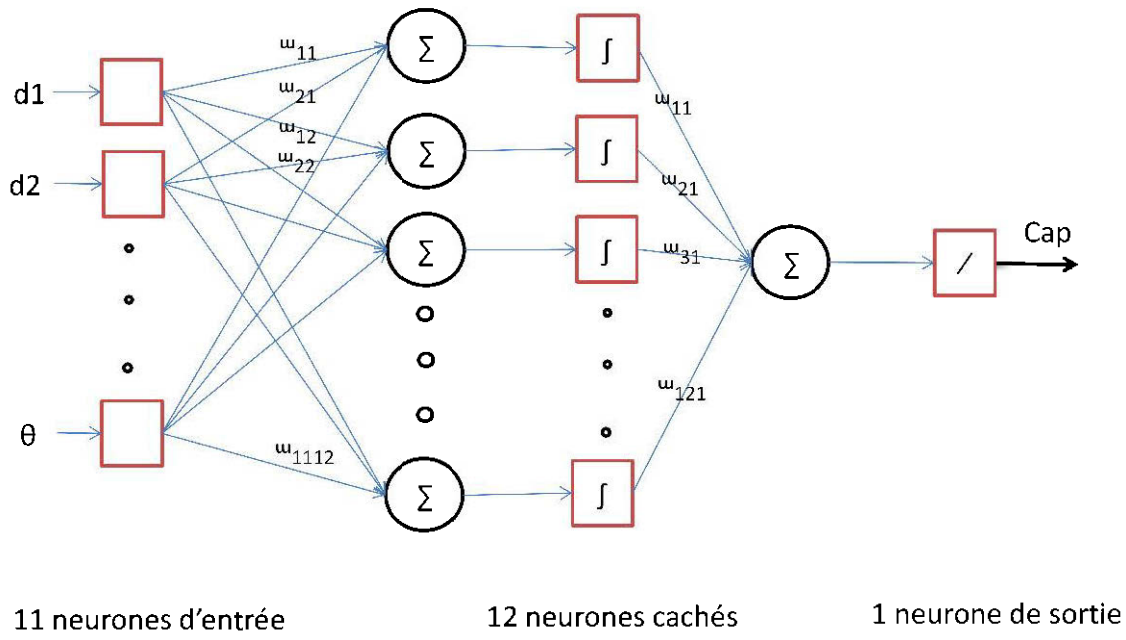


Figure 3- 12: Détail du réseau de neurones pour le contrôle latéral

3.7.2 Architecture pour le guidage longitudinal

La figure 3-13 montre le détail de l'architecture du contrôleur longitudinal. Nous avons comme entrées les cinq distances retournées par les détecteurs d'obstacles des segments du centre: d3, d4, d5, d6, d7 auxquelles est ajoutée la distance au point-cible Dp; il faut donc six neurones d'entrées. La sortie est la vitesse à adopter V; il y a donc un neurone sur la couche de sortie. La vitesse calculée est transmise au module de pilotage du robot qui agit en conséquence sur ses actionneurs de vitesse pour réaliser cette vitesse. Pour le nombre de neurones cachés, nous avons encore essayé différentes valeurs pour trouver que sept neurones cachés donnent des résultats plus précis.

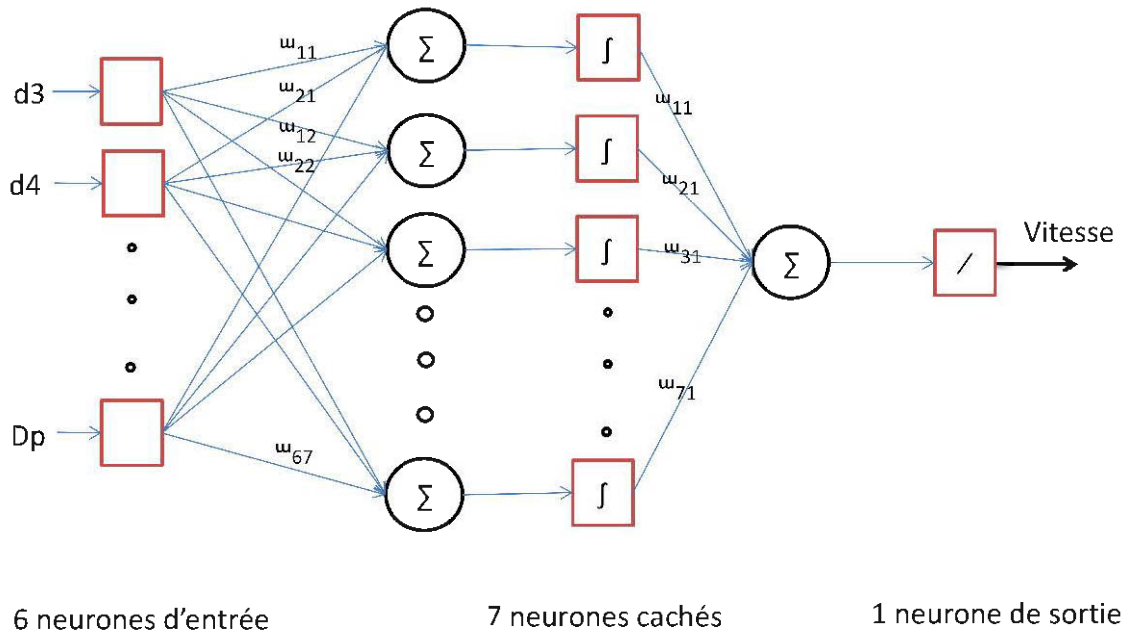


Figure 3- 13: Détail du réseau de neurones pour le contrôle longitudinal

3.8 L'apprentissage

Une phase cruciale de l'utilisation d'un réseau de neurones pour le contrôle d'un robot mobile est son apprentissage. On veut à travers la phase d'apprentissage que le réseau construise un modèle général de la correspondance entre les entrées et les sorties afin de pouvoir calculer la sortie de n'importe quelle entrée une fois l'apprentissage terminée. Le défi est de trouver les exemples nécessaires et suffisants à la construction d'un bon modèle. C'est-à-dire des exemples représentatifs de toutes les situations éventuelles. Si le nombre d'exemples n'est pas suffisant, l'apprentissage n'est pas suffisant pour construire un modèle général. S'il y a trop d'exemples, des redondances peuvent exister conduisant à un sur-apprentissage c'est-à-dire à la construction d'un modèle propre aux exemples appris; ce qui empêche toute possibilité de traitement correct d'un cas non appris. Si la base d'apprentissage n'est pas pertinente c'est-à-dire représentative du processus à modéliser, elle ne peut pas servir à la construction d'un modèle général.

Dans la littérature, on répertorie deux approches souvent utilisées pour déterminer des exemples:

- générer des exemples par des mises en œuvre expérimentales
- définir des exemples sur la base d'une règle (l'approche utilisée ici).

Dans la première approche basée sur des résultats d'expériences, on essaie différentes routes et différentes configurations d'obstacles. Cette approche demande des moyens expérimentaux importants et on peut

avec elle réussir au bout d'une longue période d'apprentissage à rencontrer toutes les situations pertinentes à la construction d'un modèle suffisamment proche de la réalité.

Dans la seconde approche (apprentissage à base de règle), l'apprentissage est fait à partir d'un petit nombre d'exemples. Ce type d'apprentissage se fait plus rapidement et ne nécessite pas de dispositifs expérimentaux. Cependant il n'est pas garanti qu'il soit représentatif de toutes les situations éventuelles. Au prochain chapitre, nous verrons les effets de cette approche sur les résultats obtenus.

CHAPITRE 4

MISE EN ŒUVRE ET RÉSULTATS

Dans ce chapitre, nous exposons notre démarche pour la mise au point des deux réseaux de neurones, les résultats obtenus et leur analyse . Nous finissons par une discussion sur l'apprentissage des réseaux de neurones.

4.1 Constitution de la base d'apprentissage

Nous avons construit une base d'apprentissage pour le module de guidage latéral et une autre pour le contrôleur longitudinal.

4.1.1 Base d'apprentissage pour le contrôle latéral

A la suite de nombreux tests avec une série d'exemples, nous avons constitué un échantillon de quarante cinq paires d'entrées - sorties désirées que nous avons au final jugé représentatif pour un bon apprentissage. Ces exemples couvrent équitablement les neufs secteurs surveillées par les détecteurs d'obstacles. Non seulement les exemples sélectionnés proviennent de tout l'espace d'entrée mais aussi ils sont en nombre égal entre les régions de l'espace.

4.1.2 Base d'apprentissage pour le guidage longitudinal

Comme dans le cas du contrôle latéral, nous avons essayé plusieurs groupes d'exemples pour sélectionner dix-huit exemples couvrant des cas que nous avons jugés au final pertinents à l'apprentissage.

4.2 Choix de l'architecture du réseau

Pour les deux contrôleurs, la structure **fitnet** est utilisée avec un nombre de neurones cachés spécifique à chaque cas.

4.2.1 Architecture pour le contrôle latéral

Nous avons implémenté la structure **fitnet,** un perceptron à trois couches avec onze neurones sur la couche d'entrée, un neurone sur la couche de sortie et douze neurones sur la couche cachée comme le montre la figure 4-1.

L'architecture **fitnet** est appropriée pour la régression non-linéaire (**fitting**). La fonction de transfert de la couche cachée est **tansig** et celle de la couche de sortie est **purelin**. Les neufs distances aux obstacles et les deux coordonnées du point-cible constituent les éléments du vecteur d'entrée. La sortie est le cap à suivre. Quant à la quantité de neurones de la couche cachée, le nombre douze a été retenu pour avoir donné de meilleurs résultats.

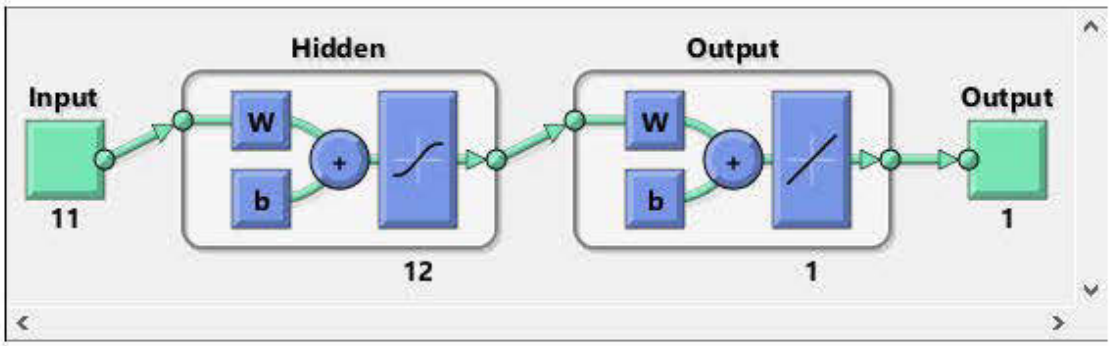


Figure 4- 1 Structure du réseau de neurones pour le contrôle du cap

4.2.2 Architecture pour le guidage longitudinal

Ici aussi, une structure **fitnet** est utilisée avec six neurones sur la couche d'entrée, un neurone sur la couche de sortie et sept neurones sur la couche cachée comme le montre la figure 4-2. Les cinq distances aux obstacles et la distance au point-cible constituent les éléments du vecteur d'entrée. La sortie est la vitesse à adopter. Quant à la quantité de neurones de la couche cachée, nous avons obtenu de meilleurs résultats avec sept neurones.

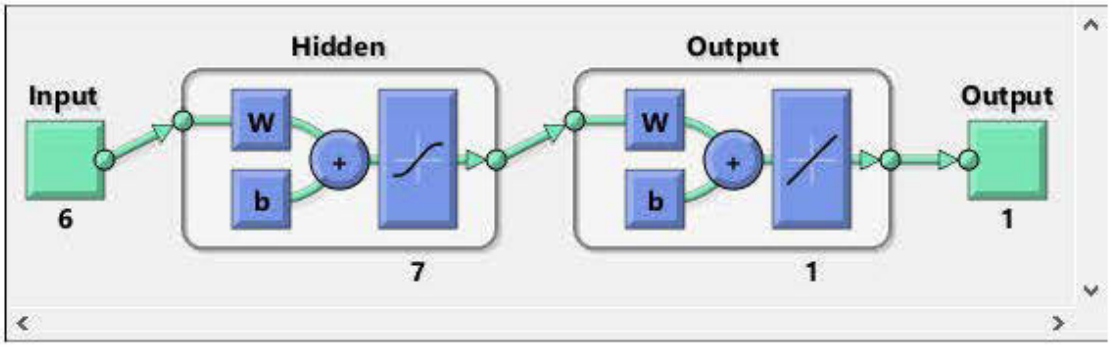


Figure 4- 2 Structure du réseau de neurones pour le contrôle de la vitesse

4.3 L'apprentissage des contrôleurs neuronaux

Les données d'entrées sont normalisées pour faciliter leur traitement par les réseaux de neurones.

Aux entrées et sorties du réseau, on associe une fonction de pré- traitement ou de conditionnement qui transforment les données d'entrée en une forme qui est plus facile ou plus efficace à traiter pour le réseau. Ici la fonction **mapminmax** est utilisée pour normaliser les données de telle sorte que toutes les valeurs tombent dans l'intervalle [-1, 1]. Cela a pour effet d'accélérer l'apprentissage. Enfin, la fonction **removeconstantrows** est utilisée pour supprimer les lignes redondantes dans les données.

L'apprentissage est fait avec la fonction **trainscg (scaled conjugate gradient backpropagation)** pour le contrôleur latéral et avec la fonction **trainlm** (Levenberg-Marquadt) pour le contrôleur longitudinal**.** Il est très difficile de savoir quel algorithme d'apprentissage sera le plus efficace pour un problème donné. Elle dépend de nombreux facteurs, notamment de la complexité du problème, du nombre d'exemples dans la base de données d'apprentissage, le nombre de poids et de biais dans le réseau, l'erreur cible recherchée et si le réseau est utilisé pour la reconnaissance de formes (analyse discriminante) ou pour l'approximation de fonction (régression). Nous avons choisi l'algorithme de descente du gradient **trainscg** (scaled conjugate gradient) pour le contrôleur latéral parce qu'il offre la meilleure performance comparativement aux autres algorithmes. Et pour le contrôleur longitudinal, l'algorithme **trainlm** (Levenberg-Marquadt) a été choisi pour la même raison.

La pratique courante en apprentissage d'un réseau multicouche est d'abord de diviser les données en trois sous-ensembles. Le premier sous-ensemble est l'ensemble d'apprentissage, qui est utilisé pour le calcul du gradient et la mise à jour des poids et des biais du réseau. Le deuxième sous-ensemble est l'ensemble de validation qui sert à évaluer la capacité à généraliser du réseau. Le troisième sous-ensemble est l'ensemble de test qui sert à évaluer la réponse du réseau aux situations nouvelles, donc pas rencontrées lors de

l'apprentissage. L'erreur sur l'ensemble de validation est contrôlé pendant le processus d'apprentissage. L'erreur de validation diminue normalement au cours de la phase initiale de l'apprentissage, comme l'erreur d'apprentissage. Toutefois, lorsque le réseau commence à «sur-apprendre», l'erreur sur la validation commence généralement à augmenter. Les poids et les biais sont enregistrés et le processus d'apprentissage est arrêté quand l'erreur sur la validation atteint son minimum. L'erreur sur le test n'influence pas l'apprentissage. Elle est utilisée pour comparer différents modèles et pour tracer l'erreur de test. Si l'erreur sur le test atteint son minimum à un nombre d'itérations significativement différent que l'erreur de validation, c'est la preuve d'une mauvaise répartition de la base de données.

Ici, pour le contrôleur latéral nous avons obtenu de meilleurs résultats en constituant une base de 45 exemples spécifiquement dédiée à l'apprentissage et une base de 16 exemples dédiée aux tests; et pour le contrôleur longitudinal, 18 exemples tous dédiés à l'apprentissage et une base de 6 exemples pour le test. Cependant nous avons utilisé dans les deux cas la fonction **dividerand** pour piquer aléatoirement les données lors de l'apprentissage.

La mesure de la qualité de l'apprentissage se fait à travers trois paramètres:

- la valeur du gradient à la fin de l'apprentissage
- l'erreur quadratique moyenne
- la régression

Durant l'apprentissage, la correction de l'erreur de prédiction consiste à modifier les paramètres qui l'engendrent au moyen de la méthode du gradient. Cette technique consiste à modifier chaque paramètre selon sa contribution à l'erreur. Dans le cas des réseaux de neurones, ce sont les valeurs des poids synaptiques qui engendrent l'erreur. On calcule le gradient de l'erreur relativement au poids de chaque neurone, puis on ajuste la valeur de chaque poids selon l'évolution du gradient qui doit décroître et devenir minimale à la fin de l'apprentissage.

L'erreur quadratique moyenne doit être la plus faible possible sans nécessairement s'annuler. Il est évident que plus elle est basse, meilleure est la prédiction.

La régression mesure la corrélation entre la sortie prédite par le réseau de neurone et la sortie désirée. Elle évalue la performance de l'apprentissage comme le pourcentage de sorties correctement prédites. Une régression de 1 indique que 100% des prédictions du réseau sont correctes. A première vue un tel résultat indique la perfection. Sauf qu'en réalité lorsque cela arrive, c'est plutôt le signe que chaque donnée a été «apprise par cœur». Ce qui dégrade la capacité de généralisation du réseau. En fait il est préférable que la régression soit comprise entre 0.9 et 0.95.

4.4 Analyse de le performance du contrôleur latéral

Dans ce paragraphe, nous discuterons de la qualité de l'apprentissage et de la capacité de prédiction du réseau de neurones pour le contrôle du cap.

4.4.1 Performance en apprentissage

a) Évolution du gradient

Le gradient, qui est parti au départ de presque 10e+5 a diminué jusqu'à atteindre le minimum de 7.14e-7 au bout de 294 itérations après quoi le processus d'apprentissage s'est arrêté.

Le nombre de test de validation ayant échoué durant l'apprentissage est marqué égal à 0 car le processus d'apprentissage n'a pas été ponctué d'épisode de validation; les 45 données de la base ayant toutes été utilisées uniquement pour l'apprentissage.

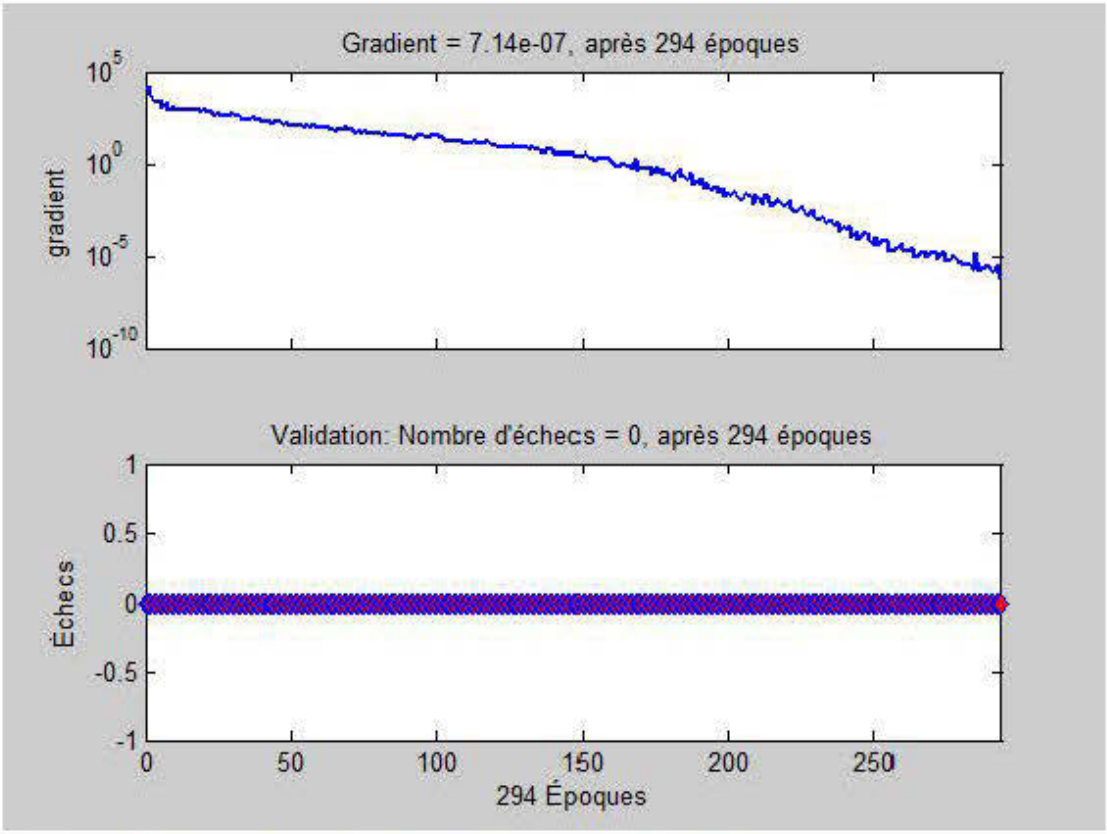


Figure 4- 3 Évolution du gradient pour l'apprentissage du contrôleur du cap

b) L'erreur quadratique moyenne

L'erreur quadratique moyenne a atteint sa valeur minimale (2.7766e-16) au bout de 294 itérations. Cette valeur est suffisamment faible pour être considérée acceptable.

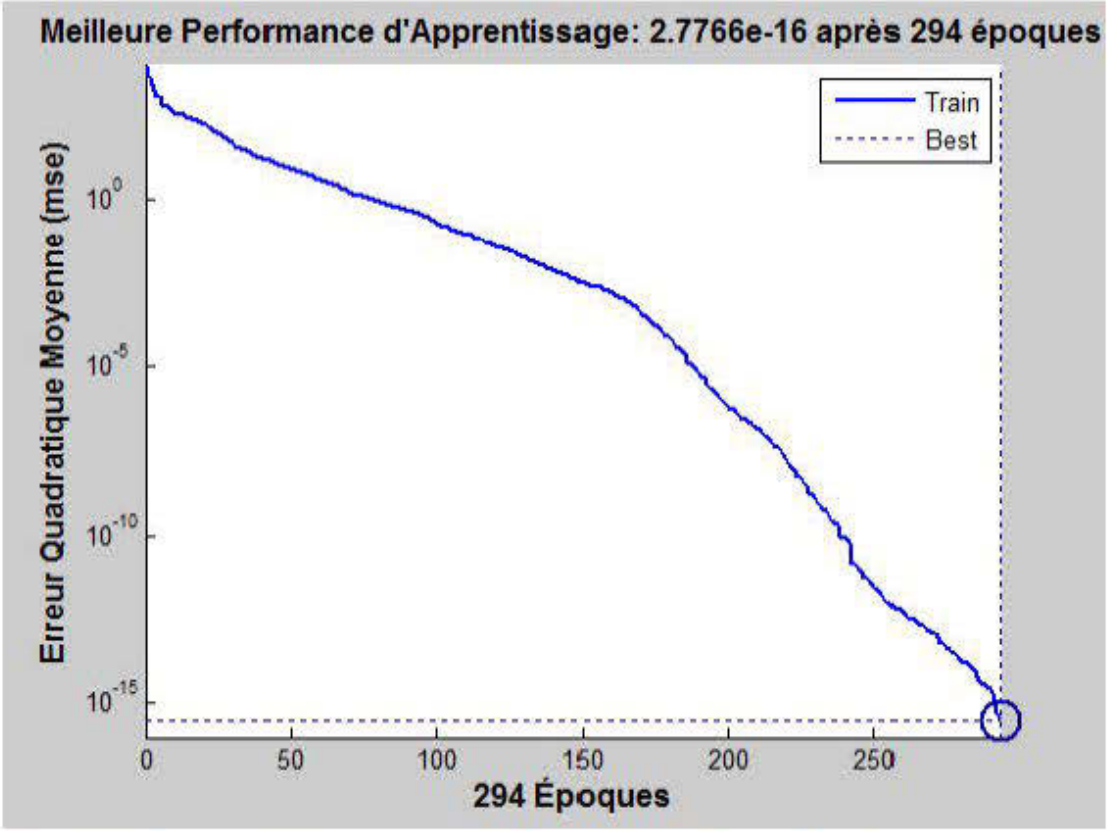


Figure 4- 4 Erreur quadratique moyenne de l'apprentissage du contrôleur de cap

c) La régression

La courbe de régression témoigne d'un apprentissage parfait (R=1) et toutes les données sont alignées. Cette perfection n'est pas forcément une bonne chose car elle dénote d'un certain sur-apprentissage nuisible à la généralisation comme nous l'avons évoqué précédemment.

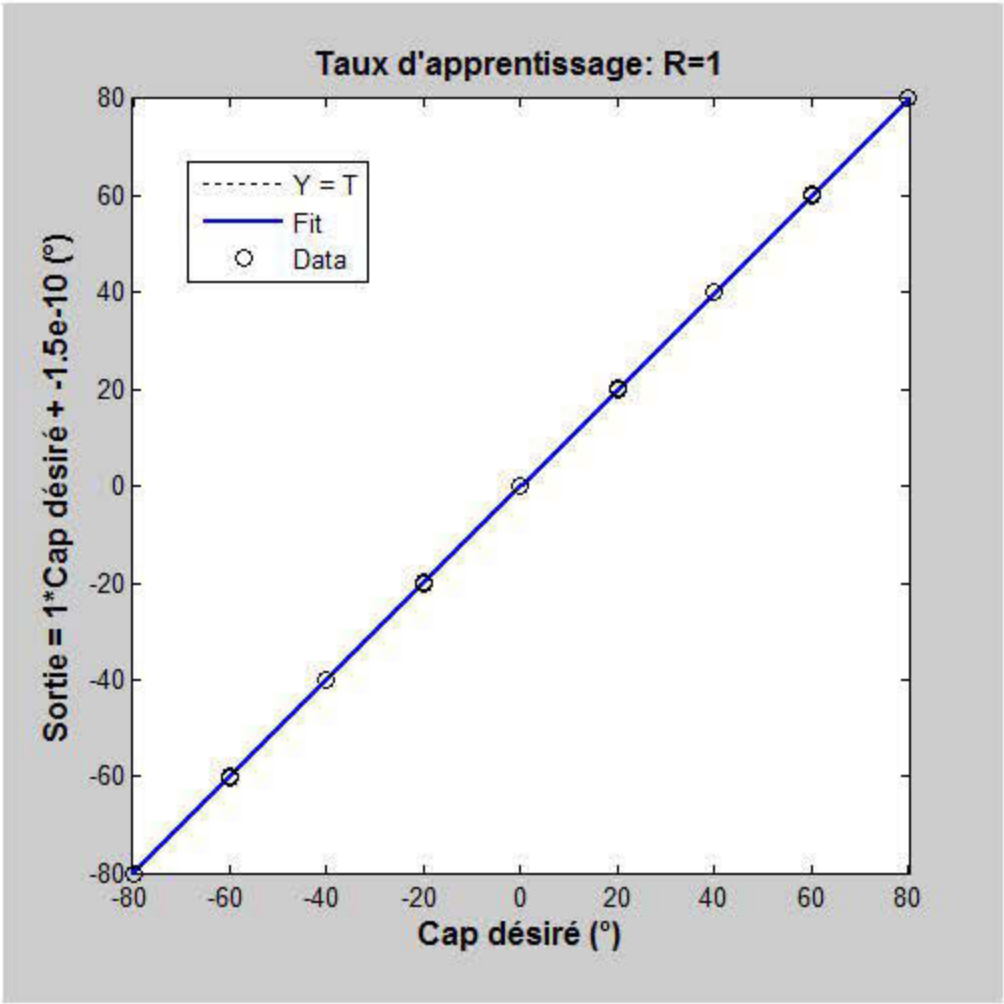


Figure 4 - 5 La courbe de la régression pour le contrôleur du cap

4.4.2 Capacité de prédiction

a) Validation avec les données de la base d'apprentissage

Nous avons fait des tests avec les données de la base d'apprentissage dans le but de vérifier si les cas appris peuvent être résolus sans erreur. Pour cela nous avons tracé la trajectoire désirée c'est-à-dire la courbe des différents angles de direction en fonction des rangs successifs des configurations d'obstacles de la base d'apprentissage. Ensuite nous avons tracé la trajectoire obtenue par le réseau de neurones c'est-à-dire la courbe des différents angles de direction donnés par le réseau de neurones en fonction des mêmes rangs successifs des configurations d'obstacles. Les deux courbes sont superposées sur la figure 4-6. Comme on peut s'y attendre (gradient très faible et courbe de régression égal à 1) la figure montre

comment la réplication des cas appris est parfaite. La trajectoire obtenue se confond point par point avec la trajectoire désirée.

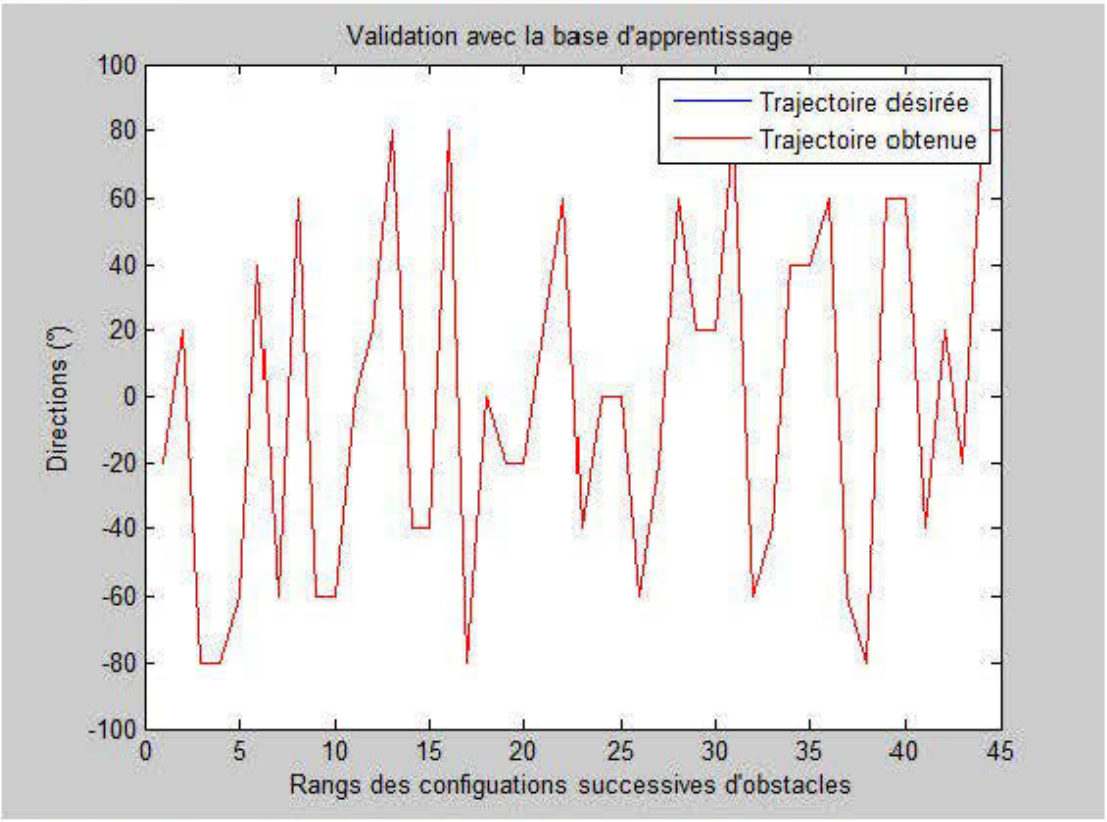


Figure 4- 6 Validation avec la base d'apprentissage du contrôleur du cap

b) Validation avec les données de la base de test

Nous avons présenté au réseau 16 exemples qui ne sont pas inclus dans la base de données d'apprentissage et comparé les réponses obtenues aux réponses désirées.

Le tableau 4.1 ci-dessous montre les exemples entrés avec les réponses obtenues et désirées correspondantes. Nous avons tracé la trajectoire désirée c'est-à-dire la courbe des différents angles de direction en fonction des rangs successifs des configurations d'obstacles de la base de test. Ensuite nous avons tracé la trajectoire obtenue par le réseau de neurones c'est-à-dire la courbe des différents angles de direction donnés par le réseau de neurones en fonction des mêmes rangs successifs des configurations d'obstacles. Les deux courbes sont superposées sur la figure 4-7. Elle montre la différence entre la trajectoire désirée et la trajectoire obtenue.

Quant à la figure 4.8, elle montre directement l'erreur de prédiction du cap c'est-à-dire la différence entre le cap désiré et le cap calculé pour les rangs successifs de configuration d'obstacles.

Chaque cap désiré est associé au centre d'un segment. Ce qui est important est que le cap calculé soit contenu à l'intérieur de ce segment. Ainsi la différence entre le cap calculé et le cap désiré ne devrait jamais dépasser 10° en valeur absolue. Dans les exemples 4, 7 et 10, cette condition n'a pas été respectée. On observe un écart de 13.2535° pour l'exemple 4, de 10.6033° pour l'exemple 7 et de 10.1802° pour l'exemple 10. Ces écarts ne provoqueraient pas forcément une collision en autant qu'ils se produisent pendant qu'on est encore loin des obstacles. En effet plus l'obstacle est proche du robot plus précis doit être la manœuvre pour son évitement. Et plus il y a d'anticipation plus on peut se permettre des écarts.

Caps désirés	-20	20	40	-60	0	20	-40	80
Caps obtenus	-19.339	21.1697	39.0868	-73.2535	-3.0811	24.7849	-50.6033	80.0209

Caps désirés	-80	20	40	-60	-20	80	-60	40
Caps obtenus	-78.623	9.8198	36.1425	-61.7153	-17.1066	82.5641	-53.7486	46.5636

Tableau 4- 1: Caps désirés et caps obtenus

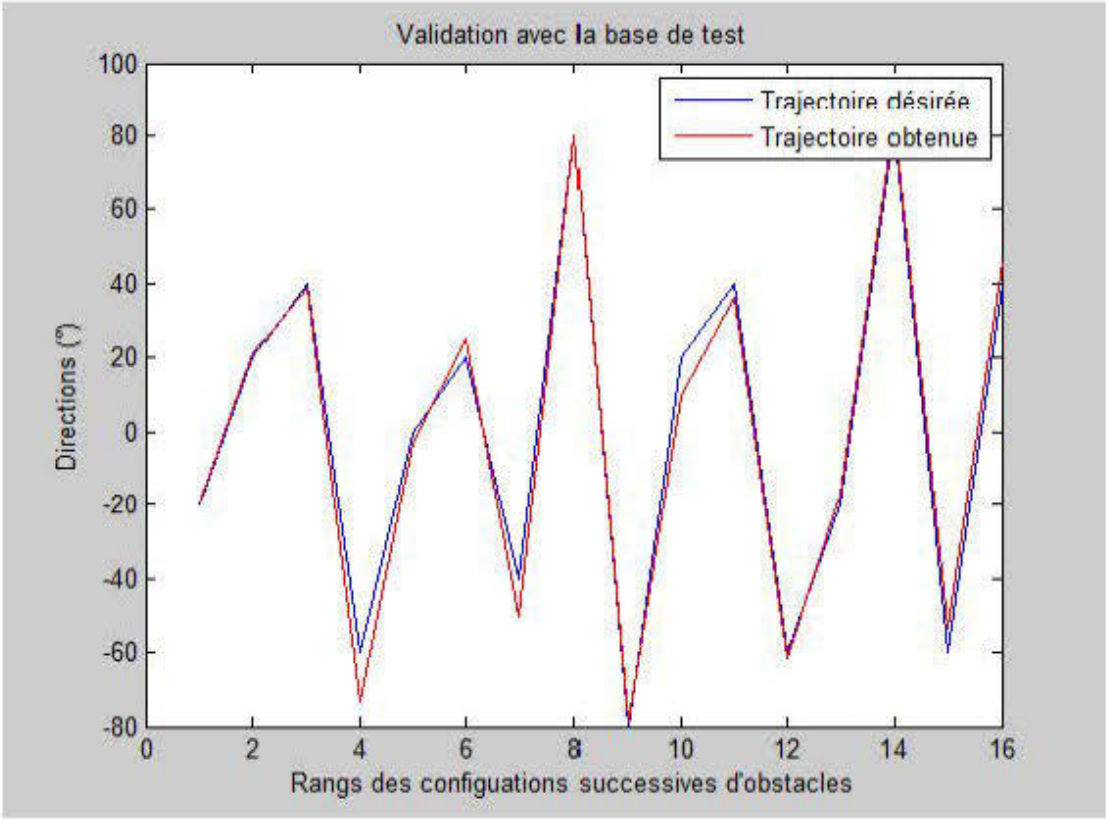


Figure 4- 7 Validation avec la base de test du contrôleur de cap

c) **Erreur de prédiction**

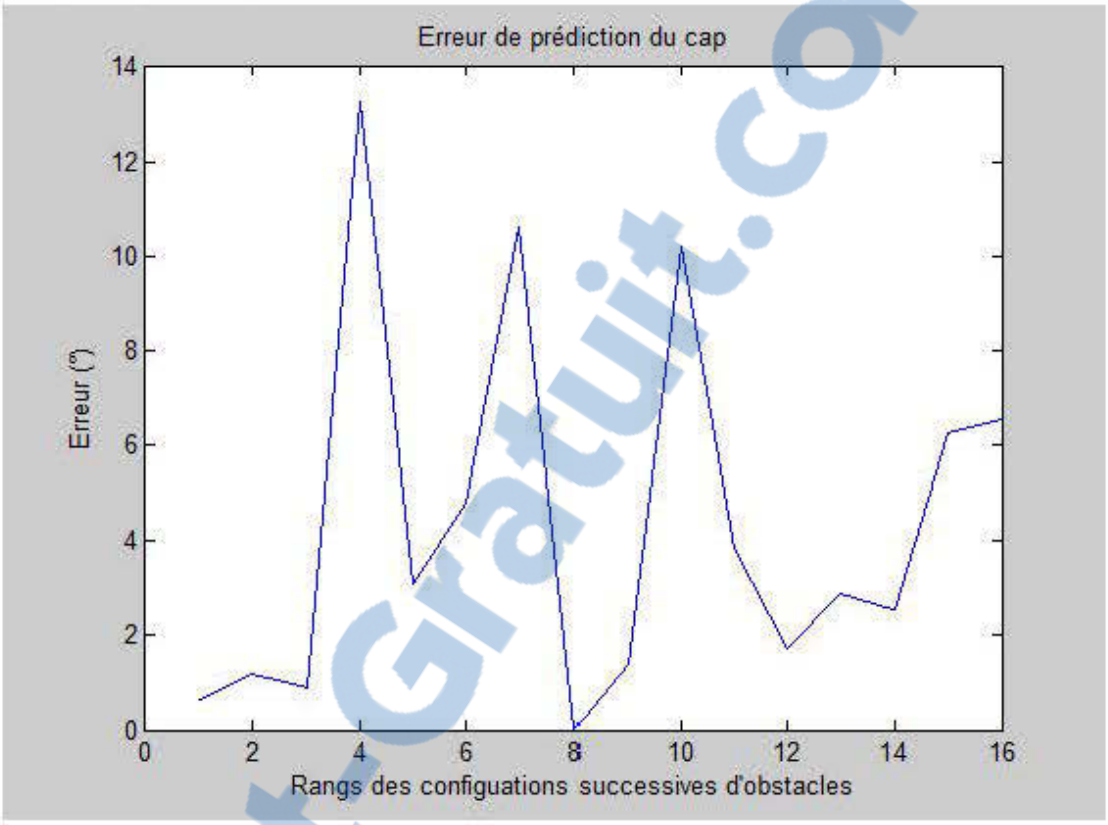


Figure 4- 8 Erreur de prédiction du cap

4.5 Analyse de la performance du contrôleur longitudinal

Ici nous discuterons de la qualité de l'apprentissage et de la capacité de prédiction du réseau de neurones pour le contrôle de la vitesse.

4.5.1 Performance en apprentissage

a) **Évolution du gradient**

Le gradient a commencé presque à 10 et a diminué jusqu'à atteindre le minimum de 1.1693e-8 au bout de 6 itérations où le processus d'apprentissage s'est arrêté.

Le nombre de test de validation ayant échoué durant l'apprentissage est marqué égal à 0 car le processus d'apprentissage n'a pas été ponctué d'épisode de validation; les 18 exemples de la base ayant tous été utilisées uniquement pour l'apprentissage.

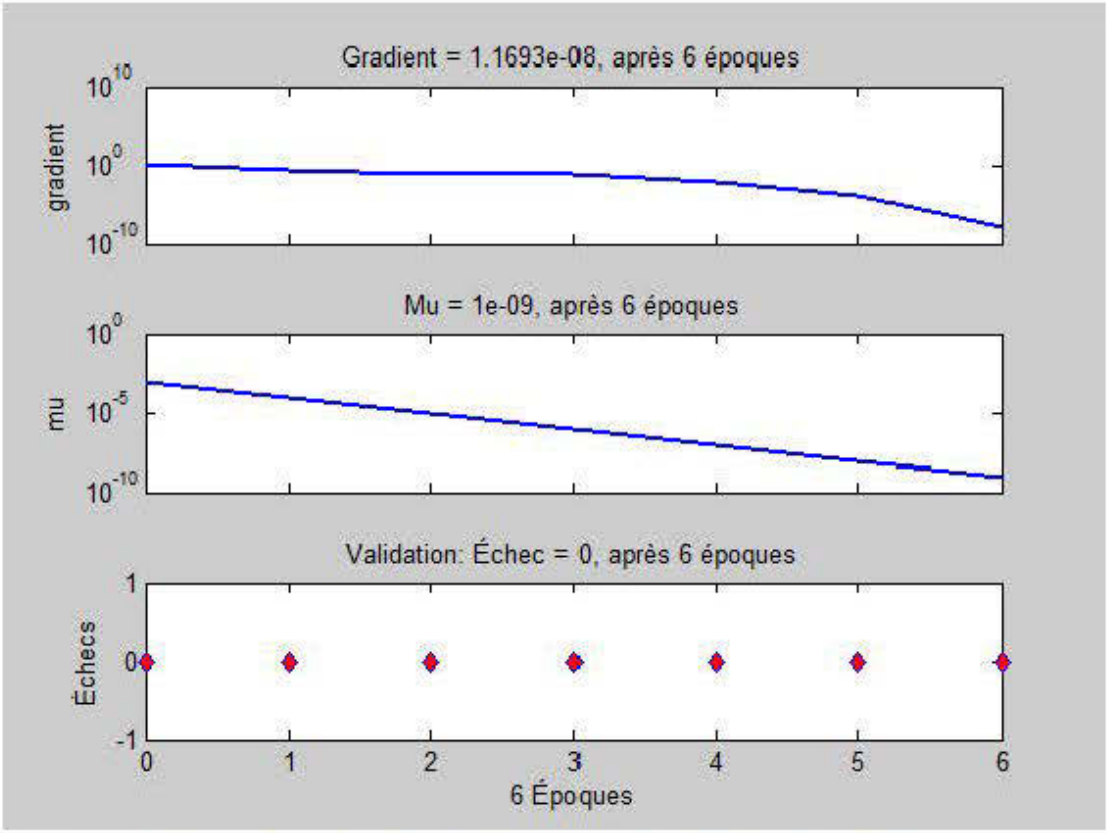


Figure 4- 9 Évolution du gradient pour l'apprentissage du contrôleur de vitesse

b) L'erreur quadratique moyenne

L'erreur quadratique moyenne a atteint sa valeur minimale (6.8657e-16) au bout de 6 itérations. Une valeur suffisamment faible pour être considérée bonne.

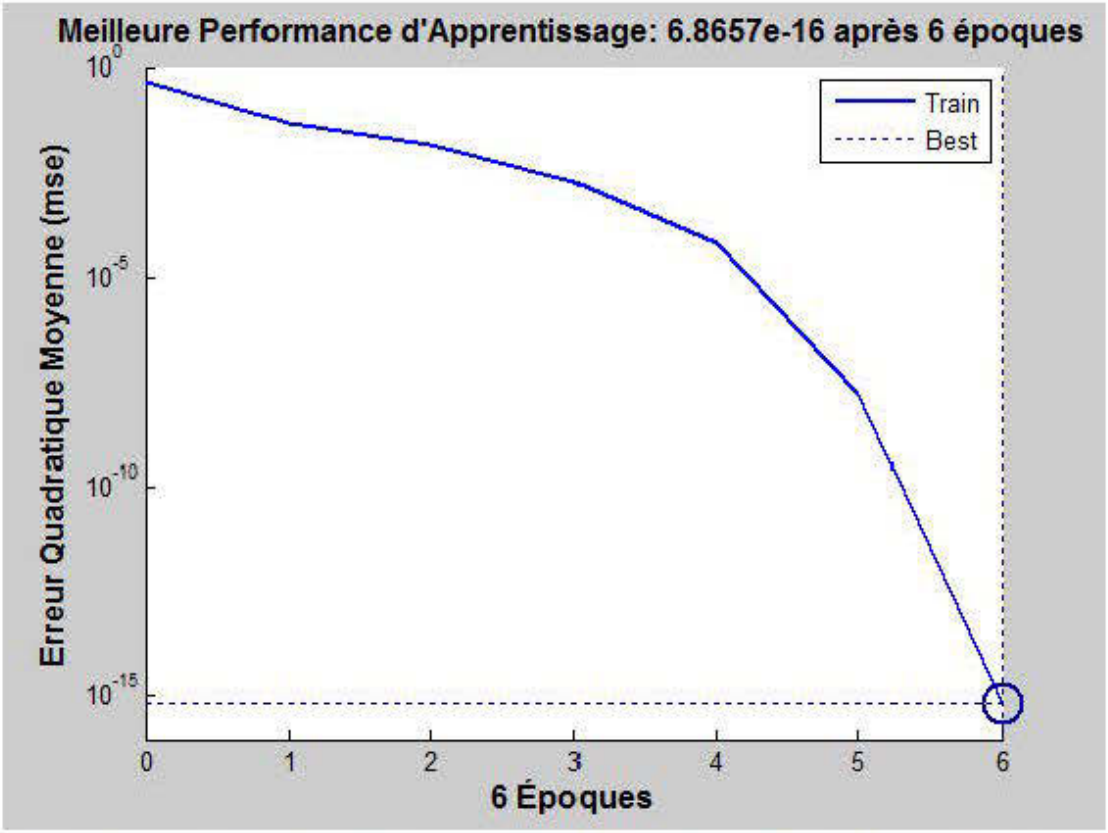


Figure 4- 10 L'erreur quadratique moyenne pour l'apprentissage du contrôleur de vitesse

c) La régression

Comme dans le cas du contrôleur latéral, la courbe de régression indique un apprentissage parfait (R=1) avec toutes les données alignées. On peut aussi y voir un certain sur-apprentissage; nuisible à la généralisation.

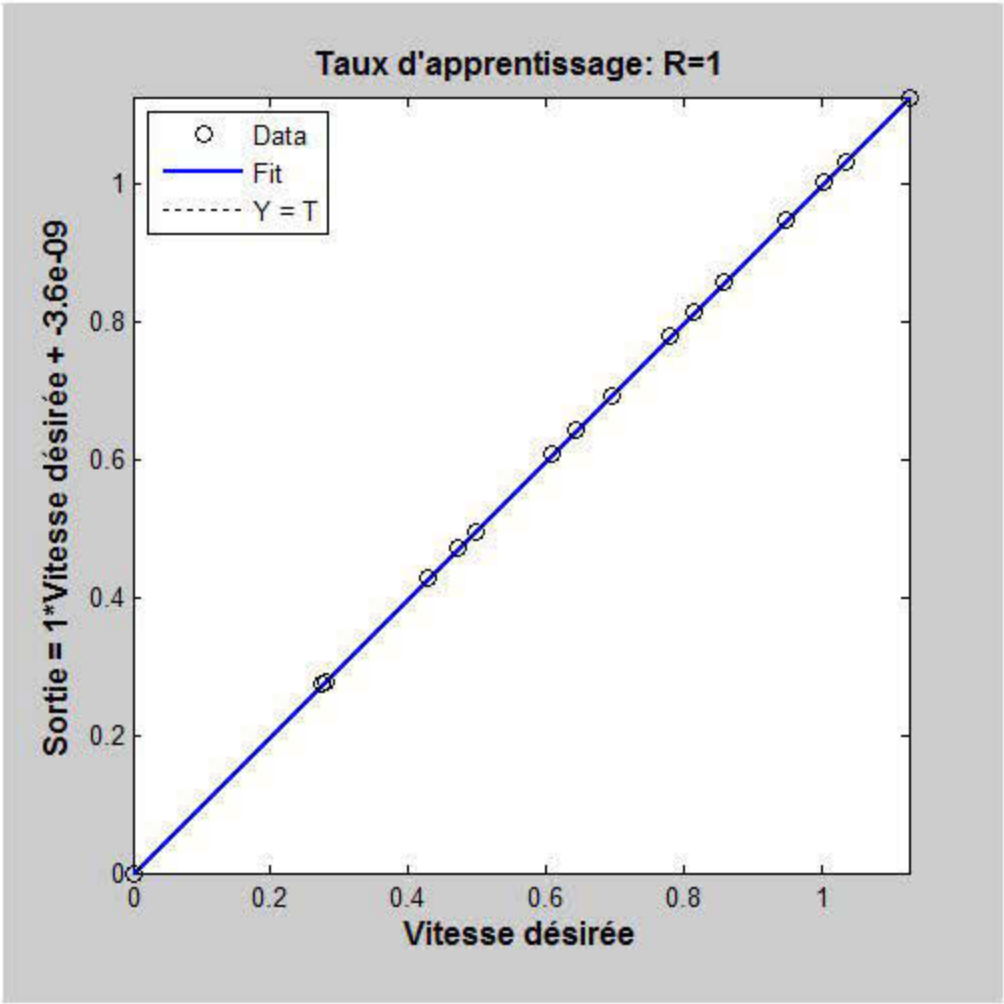


Figure 4 - 11 Courbe de régression pour le contrôleur de vitesse

4.5.2 Capacité de prédiction

a) Validation avec les données de la base d'apprentissage

Ici aussi nous avons fait des tests avec les données de la base d'apprentissage dans le but de vérifier si les cas appris peuvent être traités sans erreur. Comme dans le cas de la validation de l'apprentissage du contrôleur latéral, nous avons tracé les changements de vitesse désirés c'est-à-dire la courbe des différentes vitesses désirées en fonction des rangs successifs des configurations d'obstacles de la base d'apprentissage. Ensuite nous avons tracé les changements de vitesse donnés par le réseau de neurones c'est-à-dire la courbe des différentes vitesses calculées par le réseau de neurones en fonction des mêmes rangs successifs des configurations d'obstacles. Les deux courbes sont superposées sur la figure 4-12.

L'apprentissage ayant été adéquatement réalisé comme l'indique ses paramètres (gradient très faible et courbe de régression égal à 1) la figure montre une réplication parfaite des cas appris. Les vitesses obtenues se confondent point par point avec les vitesses désirées.

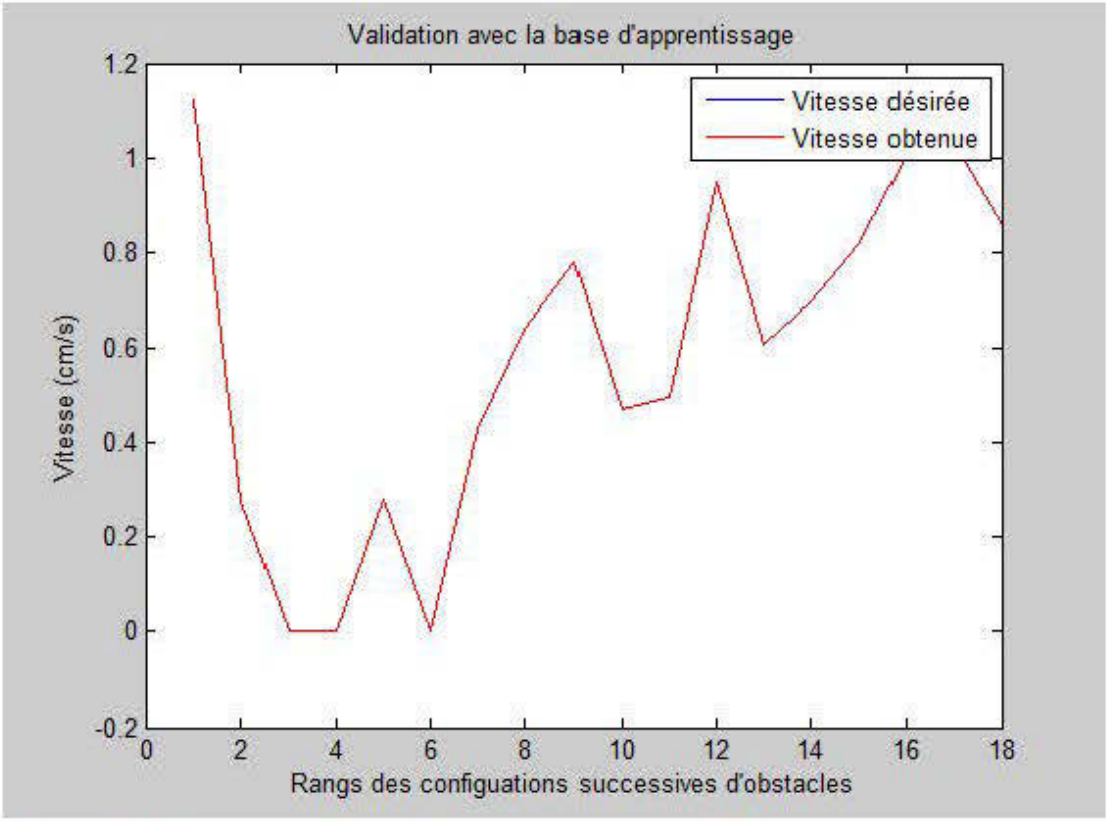


Figure 4- 12 Validation avec la base d'apprentissage du contrôleur de vitesse

b) Validation avec les données de la base de test

Nous avons présenté au réseau 6 exemples qui ne sont pas inclus dans la base des données d'apprentissage et comparé les réponses obtenues aux réponses désirées en procédant de la même façon que pour la validation avec les données de la base d'apprentissage.

Le tableau 4.2 ci-dessous montre les exemples entrés avec les réponses obtenues et désirées correspondantes.

La figure 4.13 montre la différence entre la variation des vitesses désirées et celle des vitesses calculées. La figure 4.14 montre l'erreur de prédiction de la vitesse donc la différence entre la vitesse désirée et la vitesse calculée pour les rangs successifs de configuration d'obstacles.

Les écarts observés sont acceptables sauf dans le cas où l'on désire une vitesse nulle. En effet plus la vitesse désirée est proche de la vitesse maximale plus on peut se permettre des écarts sur sa prédiction. Et plus cette vitesse est proche de 0, plus sa prédiction doit être précise pour des raisons évidentes de sécurité.

Vitesses désirées	0	0.5	2.5	1.5	2	1
Vitesses obtenues	0.256	0.5997	2.3168	1.4815	1.9308	1.0622

Tableau 4 - 2 Vitesses désirées versus vitesses obtenues

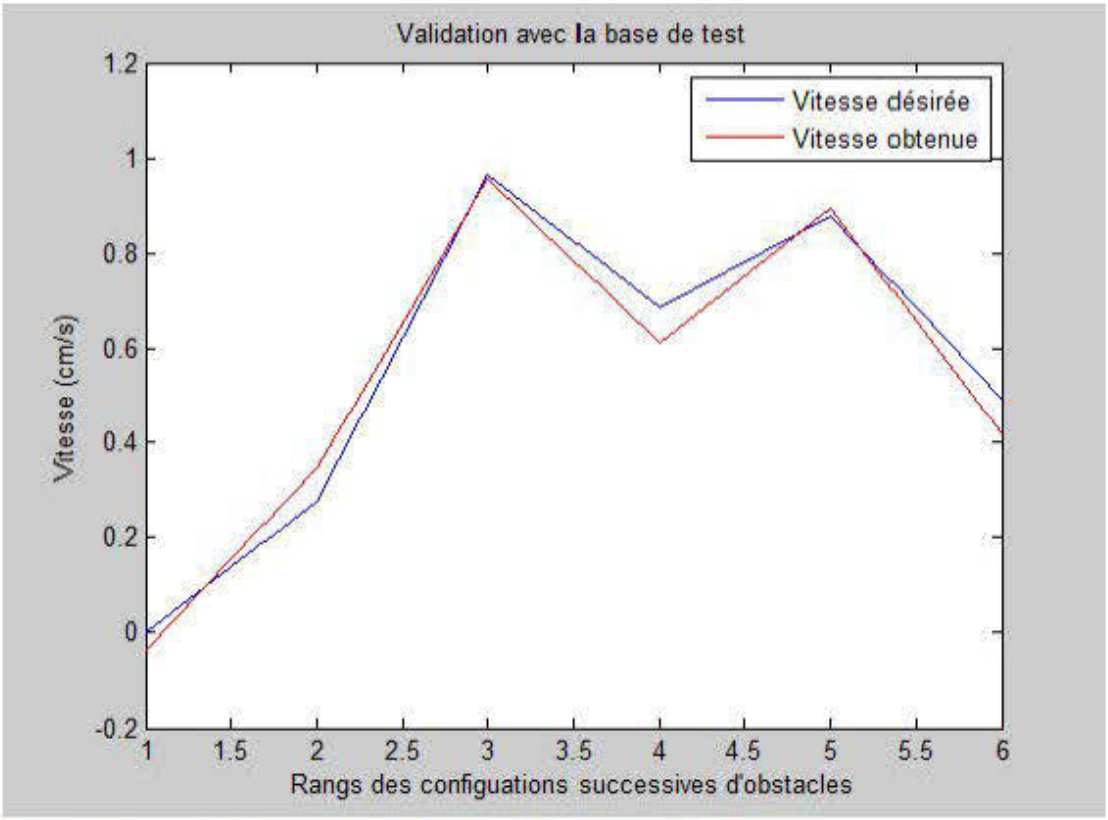


Figure 4- 13 Validation avec la base de test du contrôleur de vitesse

c) Erreur de prédiction de la vitesse

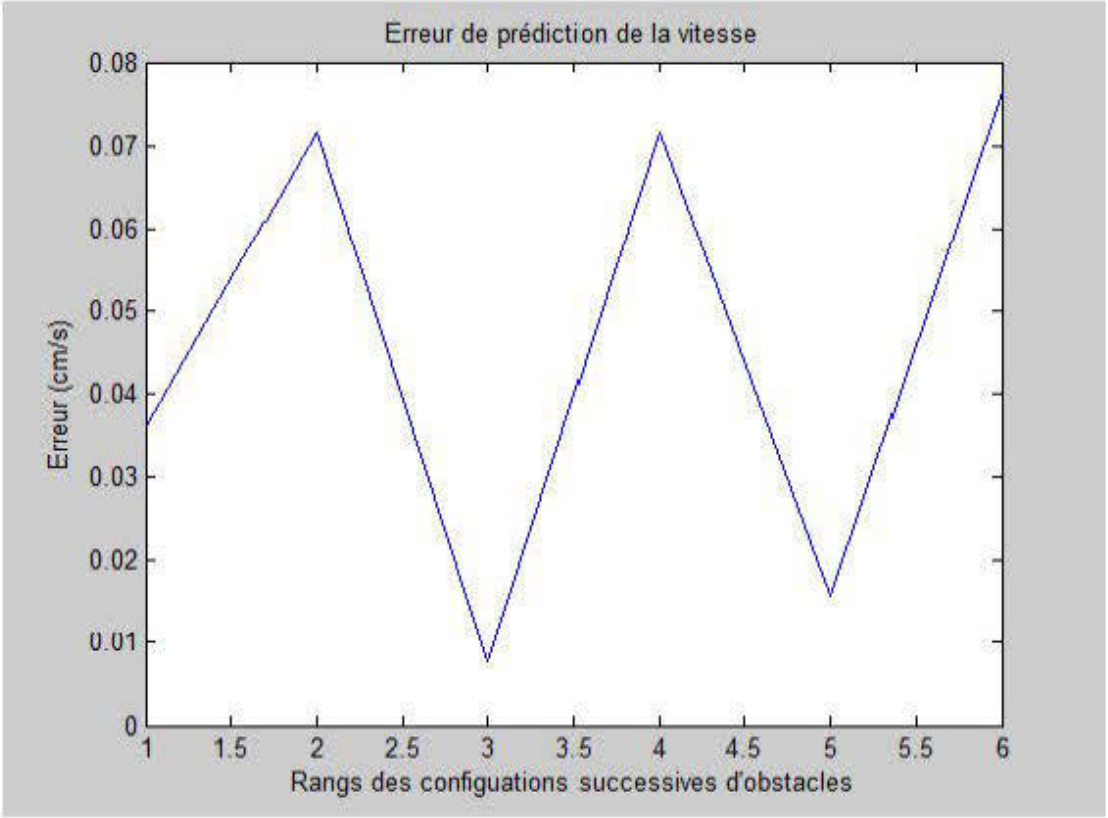


Figure 4- 14 Erreur de prédiction de la vitesse

4.6 Discussion

Comme on peut le constater, les performances des deux réseaux de neurones sont encourageants. Dans le cas du contrôleur neuronal pour la commande du cap, il est important que la marge d'erreur sur la direction calculée ne dépasse pas $\pm 10°$ sinon on déborde des limites du segment visé et cela peut être problématique surtout dans les situations de manœuvre à proximité des obstacles.

Quant à la vitesse, si la précision que l'on a ici est suffisante pour les grandes vitesses, elle est mauvaise pour les petites vitesses. Lorsqu'une vitesse proche de zéro est désirée, c'est que l'on est très proche d'un

obstacle et donc qu'on a un risque de collision élevée si on se permet des écarts dans le contrôle de la vitesse.

En fait les résultats obtenus dans ce travail souffrent des «malédictions» inhérentes aux réseaux de neurones. Elles ont pour nom: le sur-apprentissage souvent fatal, la capacité d'extrapolation souvent limitée et la difficulté de trouver une base de données appropriée pour l'apprentissage.

1) **Sur-apprentissage**

Le sur-apprentissage ou sur-ajustement (en anglais « overfitting ») est en général provoqué par un mauvais dimensionnement de la structure neuronale. De par sa trop grande capacité à stocker des informations, un réseau de neurones peut apprendre tous les détails que porte l'exemple qu'on lui présente même les plus subtils. Il se comporte dès lors comme une table ayant stockés tous les exemples présentés lors de l'apprentissage et perd ses pouvoirs de prédiction sur les nouveaux exemples.

En effet, si l'on considère un ensemble d'apprentissage et une fonction de coût quadratique, en vertu de la propriété d'approximation universelle des réseaux de neurones, il est toujours possible d'obtenir une fonction de coût aussi petite que l'on veut sur l'ensemble d'apprentissage, à condition de mettre suffisamment de neurones cachés. Cependant, le but de l'apprentissage n'est pas d'apprendre exactement la base d'apprentissage, mais le modèle sous-jacent qui a servi à engendrer les données. Or, si la fonction apprise par le réseau de neurones est ajustée trop finement aux données, elle apprend les particularités de la base d'apprentissage au détriment du modèle sous-jacent : le réseau de neurones est sur-ajusté.

Le sur-ajustement est souvent expliqué grâce aux concepts de biais et de variance. Il faut comprendre qu'en apprentissage de réseau de neurones, chaque base d'apprentissage définit un modèle. Pour un même processus dont on recherche le modèle sous-jacent au moyen d'un réseau de neurones, le réseau de neurones génère un modèle différent pour chaque base d'apprentissage.

Si l'on considère plusieurs bases d'apprentissage, le biais rend compte de la différence moyenne entre les modèles. Le biais est donc lié au niveau de bruit du processus à modéliser. La variance rend compte des différences entre les modèles selon les bases d'apprentissage.

On parle souvent de compromis entre le biais et la variance. Si un modèle est trop simple par rapport au processus à modéliser, son biais va être élevé, mais sa variance va être faible puisqu'il est peu influencé par les données. Si un modèle est trop complexe, son biais va être faible puisqu'il est capable de s'ajuster exactement à la base d'apprentissage, mais sa variance va être élevée puisqu'une nouvelle base avec une

réalisation différente du bruit peut entraîner un modèle très différent : c'est ce qui amène le sur-apprentissage.

Dans le cas d'une régression, les données de la base d'apprentissage sont bruitées (données estimées). Donc, si le modèle possède trop de degrés de liberté, il peut s'ajuster localement à certains points, et apprendre la réalisation particulière du bruit sur la base d'apprentissage et non le processus lui-même.

2) **La capacité d'extrapolation**

Les réseaux de neurones ont une tendance fâcheuse de commettre des erreurs élevées dans la prédiction des grandes variations ou des petites variations. Aussi ils sont incapables de simuler les pics de variations qui excédent le maximum ou les creux de variations qui n'atteignent pas le minimum contenu dans la série d'entrainement. Ils ne contribuent pas non plus à rendre compte des mécanismes sous-jacents au processus qu'ils modélisent. Tout ceci traduit l'incapacité des réseaux de neurones à faire des extrapolations.

Pour pallier à cette insuffisance plusieurs auteurs ont fait des suggestions au nombre desquelles:

1. Inclure dans les données d'entrainement les plus grandes variations possibles et les plus petites variations possibles.
2. Prendre le logarithme des valeurs mesurées pour réduire l'écart entre les fortes et les faibles valeurs.
3. S'assurer que les données d'entrainement contiennent suffisamment d'évènements exceptionnels pour améliorer la capacité d'extrapolation.
4. Construire des méthodes d'estimation des événements extrémaux et les inclure dans la série d'entrainement.
5. Utiliser de nouvelles fonctions d'activation
6. Élargir le plus possible l'espace des données d'entrainement

Ces différents suggestions ont rarement amélioré de façon significative la capacité d'extrapolation des réseaux de neurones. En réalité, ce qui est considéré comme une insuffisance des réseaux de neurones dérive de la nature intrinsèque de l'apprentissage statistique. En effet, on remarque que les réseaux de neurones sont efficaces lorsqu'il s'agit de faire des prévisions dans l'espace des données qui ont servi à leur apprentissage. En dehors de cet espace, leur précision reste limitée.

Pour cela dans une application de réseau de neurones, une grande attention doit être plutôt portée sur l'espace des données d'entraînement que sur l'apprentissage lui-même.

Pour une étude de la navigation autonome qui ne se rapporte pas à une manœuvre ou un comportement en particulier, qui se veut et doit être fonctionnelle pour n'importe quelle manœuvre d'évitement d'obstacles et de suivi de route dans n'importe quel environnement il y a nécessité de trouver une base de données particulièrement appropriée. On entend par base de données appropriée celle qui offre une très grande diversité en terme de perceptions-réactions.

3) **Base de données appropriée pour l'apprentissage**

Imaginez que vous voulez enseigner à vos étudiants comment résoudre n'importe quelle équation du second degré ($Ax^2 + Bx + C = 0$ avec $A \neq 0$) uniquement en donnant quelques exemples que vous auriez jugés nécessaires et suffisantes pour la cause. Votre intention est que ces étudiants découvrent à la fin de leur apprentissage que toute équation du second degré admet deux solutions distinctes ou identiques X1 et X2 telle que $X1 = \frac{-B+\sqrt{\Delta}}{2A}$ et $X2 = \frac{-B-\sqrt{\Delta}}{2A}$ avec Δ le discriminant ($\Delta = B^2 - 4AC$). Alors vous décidez de donner quelques exemples avec $B \neq 0$ et $C \neq 0$; quelques exemples avec $B = 0$ et $C \neq 0$; quelques exemples avec $B \neq 0$ et $C = 0$ et quelques exemples avec $B = 0$ et $C = 0$. A la fin de l'apprentissage vous décidez de faire un contrôle des connaissances et donner à résoudre le problème: $2X^2 + 2X + 2 = 0$. Mais comme au cours de l'apprentissage vous n'avez donné aucun exemple avec tous les coefficients identiques, certains étudiants ne réussissent pas à faire le lien entre ce cas et ceux qu'ils ont précédemment appris et donnent des réponses fausses. Vous reprenez alors l'apprentissage en ajoutant des exemples avec les trois coefficients identiques. Et à la fin de cette séance, vous décidez de faire un autre contrôle des connaissances et donner à résoudre le problème: $5X + 3X^2 + 1 = 0$. Et comme durant l'apprentissage, dans tous les exemples que vous avez donnés, les termes ont été toujours disposés dans l'ordre $Ax^2 + Bx + C$, certains étudiants y voient encore un cas sans lien avec ce qu'ils ont appris jusque là et donnent des réponses fausses. Alors vous décidez d'ajouter d'autres exemples à l'apprentissage avec les termes ordonnés de toutes les manières possibles. A la fin de la séance, vous décidez de faire un contrôle des connaissances en donnant à résoudre le problème: $-X^2 - 2X - 4 = 0$. Cependant durant les séances d'apprentissage, vous n'avez donné aucun exemple où tous les coefficients sont négatifs. Alors certains étudiants ne voient pas de lien entre les cas appris et le cas de l'examen et donnent des réponses fausses. C'est comme ça que fonctionnent les réseaux de neurones.

Mais que s'est-il passé? Le fait que tous les coefficients soient tous différents ou identiques est un bruit et n'a pas d'importance. Le fait que les termes soient disposés selon un ordre ou un autre est un bruit et n'a pas d'importance. Le fait que tous les coefficients soient positifs ou négatifs est un bruit et n'a pas

d'importance. Pourtant le réseau de neurones considère toutes les situations comme importantes et les apprend, se transformant par le fait en une simple table de stockage des exemples présentés lors de l'apprentissage plutôt qu'un modèle du processus.

Lorsque vous avez donné des exemples avec $B \neq 0$ et $C \neq 0$; des exemples avec $B = 0$ et $C \neq 0$; des exemples avec $B \neq 0$ et $C = 0$ et des exemples avec $B = 0$ et $C = 0$, vous avez dit au réseau que les valeurs prises par les coefficients B et C avaient d'importance. Pourtant ce n'est nullement votre intention. Mais malheureusement c'est ce que vous avez fait. Vous avez créé un pattern de données. Lorsque vous avez considéré que certains étudiants semblent croire que l'ordre des termes avait d'importance, et que vous avez décidé de contrer cela en ajoutant des exemples avec les termes ordonnés de toutes les manières possibles, vous avez encore malheureusement créé un autre pattern de données faisant croire que l'ordre des termes avait d'importance. Ce qui est tout l'inverse de votre intention. Le fait de donner beaucoup d'exemples de même nature ou de donner toutes les variantes d'un exemple crée un pattern que le réseau de neurones intègre dans l'ajustement de ses poids. Il faut éviter de faire ressortir les patterns dans les exemples lorsqu'ils ne comptent pas. Les réseaux de neurones sont sensibles aux patterns. Si ces derniers n'ont pas d'importance dans la construction du modèle, il faut éviter de les ressortir dans les exemples de la base d'apprentissage.

En apprentissage statistique, il est important de répondre adéquatement aux questions: quelle sont les données nécessaires et suffisantes pour représenter le modèle? Et combien de neurones cachés sont nécessaires pour les intégrer? Car les données utilisées pour l'apprentissage doivent être représentatives du processus à modéliser. C'est la mal représentativité des données qui fait le lit du sur-apprentissage ou du sous-apprentissage. A défaut de donner une méthode générale pour spécifier les données pertinentes à un apprentissage optimal, on peut identifier les pièges à éviter comme:

1. le glissement dans les données d'indices non importantes pour le modèle
2. le manque de données ou le trop plein de données
3. les redondances de données
4. les dépendances dans les données: il faut respecter une indépendance stricte entre les données

CONCLUSION

Dans ce travail, une stratégie de planification de mouvement pour la navigation autonome de robot a été proposée et implémentée au moyen d'un perceptron multicouche appris avec l'algorithme de rétro-propagation du gradient. Cette stratégie réalise un couplage adaptatif entre planification perception et action qui vise à intégrer un évitement anticipatif des obstacles; ces derniers pouvant être fixes ou mobiles. On ne vise dans le cadre de ce travail que les déplacements vers l'avant dans un plan. Cependant l'algorithme est facilement généralisable au déplacement en arrière et aux espaces à trois dimensions.

La stratégie utilise deux réseaux de neurones fonctionnant en parallèle, l'un pour la commande du cap et l'autre pour la commande de la vitesse utilisant les deux le même système perceptif. En général, dans les approches neuronales de commande de robot mobile, les réseaux de neurones sont utilisés à des fins de classification de l'environnement de déplacement en zones navigables ou non situées à droite, à gauche ou au centre qui dirige le robot vers une zone ou une autre. On tourne ainsi toujours vers la droite ou vers la gauche d'un même angle préalablement choisi. Ici les réseaux de neurones émulent la stratégie de navigation proposée et calculent directement la direction et la vitesse à adopter. Les propriétés d'approximation universelle des perceptrons multicouches attestent de la validité de cette approche. En effet si on se donne la fonction théorique que doit réaliser le réseau de neurones selon la tâche considérée (c'est-à-dire le prédicteur théorique associé à un modèle-hypothèse donné pour une tâche de modélisation), la propriété d'approximation universelle affirme que, si cette fonction existe, elle est réalisable par un réseau de neurones et l'universalité des algorithmes d'apprentissage garantit que cette réalisation est possible en pratique, quelle que soit la complexité de la fonction.

Cependant même si théoriquement un réseau de neurones multicouche peut approximer n'importe quelle fonction linéaire ou non-linéaire, ses réponses ne collent pas toujours avec les réponses désirées. Et on a souvent à gérer des écarts.

Ce qui est important est de maitriser ces erreurs c'est-à-dire les contenir dans un certain intervalle et les tolérer. Il faut faire en sorte que la réponse désirée ne soit pas la seule réponse appropriée. Car un réseau de neurones converge accidentellement vers n'importe quelle valeur appartenant au voisinage de la réponse désirée. Dans cette perspective, ce n'est pas parce que la direction ou la vitesse indiquée par le réseau de neurones ne correspond pas aux valeurs désirées que cela provoquera nécessairement un accident. Mais pour qu'il en soit ainsi, il faut que la stratégie d'évitement d'obstacles proposée pour la navigation autonome présente de fait cette caractéristique de robustesse aux erreurs du réseau de neurones

qui la modélise. Dans notre cas ici, une erreur de 5° en deçà ou au delà du cap désiré préserve la sécurité de la navigation. Bref pour tirer le meilleur d'un réseau de neurone, il faut que ses erreurs soient tolérables et tolérées.

RÉFÉRENCES

[1] R.C. Arkin. Motor schema-based mobile robot navigation. Int. J. Robot Autom., 8(4.1): 92-112. 1989

[2] C. M. Bishop. Neural Networks and their applications. Neural Computing Research Group. Aston University, Birmingham. United Kingdom. March 1994

[3] R. A. Brooks. Elephants Don't Play Chess. Robotics and Autonomous Systems 6 (1990) 3-15

[4] R. Brooks. A robust layered control system for a mobile robot. IEEE J. Robot Autom., 2: 14-23. 1986

[5] Y. C. Fai, S. H.M. Amin et Al. Bluetooth enabled mobile robot. IEEE ICIT '02. 2002 IEEE International Conference on Industrial Technology.

[6] A. Dib. Commande non-linéaire et asservissement visuel de robots autonomes. Thèse de doctorat. École Supérieure d'Électricité. Octobre 2011

[7] A. Durand Petiteville. Navigation référencée multi-capteur d'un robot mobile en environnement encombré. Thèse de doctorat. Université de Toulouse III. Janvier 2012

[8] M. Farid. Système de détection d'obstacle pour véhicule minier souterrain. Mémoire de maitrise, École polytechnique de Montréal, Novembre 1997.

[9] D. Filliat. Navigation, perception et apprentissage pour la robotique. Habilitation à diriger des Recherches. Université Pierre et Marie Curie. Juillet 2011.

[10] T. Fraichard. Contribution à la planification du mouvement. Habilitation à diriger des Recherches. Institut Polytechnique de Grenoble. Mars 2006.

[11] R. Glasius, A. Komoda, S. Gielen. A biologically inspired neural net for trajectory formation and obstacle avoidance. Biological Cybernetics. June 1994.

[12] M. Harb, R. Abielmona, E. Petriu. Speed Control of a Mobile Robot Using Neural Networks and Fuzzy Logic. Proceedings of International Joint Conference on Neural Networks. Atlanta, Georgia, USA, June 14-19, 2009

[13] Q. He. Neural Network and Its Application in IR, 1999

[14] M. Hudson Beale, M. T. Hagan, H. B. Demuth.
Neural Network Toolbox. User's Guide, R2013b

[15] D. Janglová, Neural Networks in Mobile Robot Motion, pp. 15-22,
Inernational Journal of Advanced Robotic Systems, Volume 1 Number 1 (2004),

[16] A.K. Jain, K. Mao & K. M. Mohiuddin. Artificial Neural Networks: A Tutorial (1996)

[17] B. Jincun, L. Qi et al. The design of recue robot long distance control based on 3G and GPS. International Conference Intelligent Human-Machine Systems and Cybernetics. 2009

[18] M. M. Joshi, M. A. Zaveri. Optimally Learnt, Neural Network Based Autonomous Mobile Robot Navigation System. ACEEE Int. J. on Control System and Instrumentation, Vol. 02, No. 01, Feb 2011.

[19] M. M. Joshi, M. A. Zaveri. Reactive Navigation of Autonomous Mobile Robot Using Neuro-Fuzzy System. International Journal of Robotics and Automation (IJRA), Volume (2) : Issue (3), 2011

[20] S. Kahar, R. Sulaiman et Al. A Review of Wireless Technology Usage for Mobile Robot Controller. 2012 International Conference on System Engineering and Modeling.

(ICSEM 2012) IPCSIT vol. 34, Singapore

[21] S. Kahar, R. Sulaiman et Al. Data Transferring Technique for Mobile Robot Controller via Mobile Technology. 2011 International Conference on Pattern Analysis and Intelligent Robotics, 28-29 June 2011, Putrajaya, Malaysia

[22] J. Laneurit. Perception multisensorielle pour la localisation d'un robot mobile en environnement extérieur, application aux véhicules routiers. Thèse de doctorat. Université Blaise Pascal- Clermont II. Juillet 2006

[23] O. Lefevre. Navigation autonome sans collision pour robots mobiles non-holonomes. Thèse de doctorat. Institut Polytechnique de Toulouse Juillet 2006.

[24] P. K. Mohanty, D. R. Parhi. Controlling the Motion of an Autonomous Mobile Robot Using Various Techniques: a Review. Journal of Advance Mechanical Engineering. Published online 27 October 2012

[25] A. Murray, C. Odhams. Identification of driver steering and speed control. PHD dissertation. Cambridge University. September 2006.

[26] D. Nakhaeinia, S. H. Tang, S. B. Mohd Noor, and O. Motlagh. A review of control architectures for autonomous navigation of mobile robots. International Journal of the Physical Sciences Vol. 6(2.1), pp. 169-174, 18 January, 2011

[27] Y. Negishi, J. Miura et Al. Vision-Based Mobile Robot Speed Control Using a Probabilistic Occupancy Map. Conf. on Multisensor Fusion and Integration in Intelligent Systems. PP 63-69, Tokyo, July 2003.

[28] M. Parizeau. Réseaux de neurones. Cours Université Laval. Automne 2004

[29] D. Partouche. Intelligent Speed Adaptation in curves for Autonomous Vehicles. September, 2006.

[30] I. Rivals. Modélisation et commande de processus par réseaux de neurones ; application au pilotage
d'un véhicule autonome. Thèse de doctorat. Université Pierre et Marie Curie 1995

[31] J. M. Roberts, E. S. Duff, P. I. Corke. Reactive navigation and opportunistic localization for autonomous underground mining vehicles. Elsevier. November 2001.

[32] J. Sfeir. Navigation d'un robot mobile en environnement inconnu utilisant les champs de potentiels artificiels. Mémoire de maitrise. École de Technologie Supérieure. Montréal, 26 Novembre 2009

[33] P.S. Sastry, G. Santharam, K.P. Unnikrishnan. Memory neuron networks for identification and control of dynamical systems. IEEE transactions on neural networks. Vol.5 No2. March 1994

[34] M. K. Singh, D. R. Parhi. Intelligent Neuro-Controller for Navigation of Mobile Robot. ICAC3'09, January 23–24, 2009, Mumbai, Maharashtra, India.

[35] M.K. Singh, D. R. Parhi. Intelligent Neuro-Controller for Navigation of Mobile Robot. 2009 Proceedings of the International Conf. on Advances in Computing, Communication and Control

[36] M. Taix. Contribution à la planification de mouvement en robotique. Habilitation à diriger des Recherches. Université Paul Sabatier, Toulouse III. Mars 2011.

[37] G. N. Tripathi, V.Rihani. Motion planning of an autonomous mobile robot using artificial neural network. 2012

[38] J. Velagic, N. Osmic, B. Lacevic. Neural network controller for mobile robot motion control.

World academy of science. Engineering and Technology 2008.

[39] S. X. Yang, M. Meng. Real-time Collision-free Path Planning and Tracking Control of a Nonholonomic Mobile Robot using a Biologically Inspired Approach. Proceeding of MO1 IEEE International Symposium on Robotics and Automation 2001 Alberta Canada

[40] H. Yonaba. Modélisation hydrologique hybride: réseaux de neurones - modèle conceptuel. Thèse de doctorat, Université Laval, Québec 2009.

[41] T. Zhao, Y. Wang. A Neural-Network Based Autonomous Navigation System Using Mobile Robots. 12th International Conference on Control, Automation, Robotics & Vision Guangzhou, China, 5-7th December 2012 (ICARCV 2012)

[42] J. Zilkova, J. Timbo, P. Girovsky. Non linear system control using neural network. Department of Electrical Drives and Mechatronics. Technical University of Kosice 2006.