CONCEPTION DE STRUCTURES NEURONALES POUR LE CONTRÔLE DE ROBOTS MOBILES AUTONOMES

THÈSE N° 1598 (1997)

PRÉSENTÉE AU DÉPARTEMENT D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Francesco MONDADA

Ingénieur en microtechnique diplômé EPF

originaire de Minusio (TI)

acceptée sur proposition du jury:

Prof. J.-D. Nicoud, directeur de thèse
Dr P. Bessière, corapporteur
Dr S. Nolfi, corapporteur
Prof. R. Siegwart, corapporteur

Lausanne, EPFL
1997

Résumé

Les applications de la robotique mobile autonome pourraient être nombreuses: le robot facteur, le robot aspirateur domestique ou nettoyeur industriel, le robot surveillant, le robot démineur et bien d'autres applications ont un marché considérable. Malgré le marché potentiel et les possibilités technologiques actuelles, seules quelques applications très limitées sont exploitées commercialement, ce qui prouve la difficulté de ces développements du point de vue intelligence.

Face à l'échec des méthodes de conception classiques liées à l'intelligence artificielle, plusieurs nouvelles approches ont été proposées. Parmi celle-ci on trouve les réseaux de neurones et, plus récemment, les algorithmes génétiques, qui ont émergé de l'intérêt suscité par la *vie artificielle* (*Artificial Life*). L'application de ces techniques aux robots mobiles a été étudiée principalement en simulation, et dans la plupart des cas sans validation sur des robots réels. L'implémentation sur un robot mobile physique pose en effet souvent des problèmes pratiques qui limitent fortement leur utilisation.

Ce travail a permis d'analyser plusieurs structures neuronales utilisées pour le contrôle d'un robot mobile, et ceci entièrement sur un robot physique. Pour atteindre ce but, une partie de l'effort a été consacrée à la mise au point d'outils matériels et logiciels. La collaboration avec Edo Franzi, André Guignard et Yves Cheneval a permis de développer le robot mobile Khepera et ses accessoires, ainsi que le logiciel de contrôle associé.

Ces outils ont permis de réaliser six expériences qui couvrent le spectre des possibilités d'utilisation des réseaux de neurones pour le contrôle d'un robot mobile autonome. Ces expériences se limitent toutefois à la réalisation de comportements simples réalisés avec des réseaux de petite taille.

Les deux premières expériences permettent, par un comportement conçu manuellement et finement ajusté, de mettre en évidence le rôle des interactions entre le robot et son environnement. La tâche exécutée est collective dans la première expérience, et coopérative dans la deuxième.

Afin de permettre une adaptation du robot à son environnement, la troisième expérience introduit l'utilisation d'algorithmes d'apprentissage. Avec cette technique, un système capable d'apprendre à utiliser de la vision artificielle pour effectuer de l'évitement d'obstacles a été réalisé. Cette approche se montre toutefois très limitée à cause des structures très rigides nécessaires à son fonctionnement.

Les dernières trois expériences mettent en oeuvre les algorithmes génétiques, qui se révèlent être un mécanisme d'adaptation plus flexible. La première de ces trois expériences prouve la faisabilité de l'approche, tandis que la deuxième va plus en profondeur, en essayant de mieux exploiter les caractéristiques des algorithmes génétiques. Finalement algorithmes génétiques et apprentissage sont combinés dans la dernière expérience, donnant lieu à un système de contrôle d'un degré d'adaptation élevé.

L'analyse des résultats a permis de mettre en évidence avantages et inconvénients de chaque approche dans l'optique d'une utilisation concrète, et ceci afin de permettre une meilleure orientation de la recherche dans ce domaine.

Abstract

There is a large number of possible applications in the field of mobile robotics: Mail delivery robots, domestic or industrial vacuum cleaners, surveillance robots, demining robots and many others could be very interesting products. Despite this potential market and the actual technology, only few simple systems are commercially available. This proves that there are several important and problematic issues in this field, mainly at the intelligence level.

As a reaction to the failure of the classical artificial intelligence applied to the field of mobile robotics, several new approaches have been proposed. Artificial neural networks are one of these, and genetic algorithms, supported by the *Artificial Life* trend, are also getting more and more consideration. These two techniques have already been applied to mobile robotics, but mainly in simulation, and without a final test on a real mobile robot. The use of physical robots for this research seems to be still problematic due to the lack of efficient tools.

Several neural structures for the control of mobile robots have been analysed in this work. All experiences have been carried out on physical robots. To reach this goal, an important effort has been made in order to design new efficient robotic tools. Together with Edo Franzi, André Guignard and Yves Cheneval, we have developed and built hardware and software tools that make an efficient research work possible. Along with several analysis software tools, the mobile robot Khepera has been a result of this development.

Using this equipment, six experiences have been carried out, covering a large spectrum of the possible ways neural networks can be used for the control of mobile robots. These experiments have nevertheless been restricted to simple behaviours and small neural networks.

The first two experiments show, with a very simple and manually adjusted behaviour, the important role of the interaction of the robot with its environment. The first experiment is based on a collective behaviour, the second on a collaborative one.

The adaptation of the robot to the environment is introduced in the third experiments, in which a learning technique is applied. The result is a robot able to learn how to use visual stimuli to avoid particular obstacles. Despite its interesting results, this approach has turned out to be very limited, due to the rigid structure needed.

The last three experiments demonstrate the possibilities of the use of genetic algorithms, which proved to be a very flexible adaptation mechanism. The first of these three experiments tests the feasibility of this approach. The second one takes advantage of the characteristics of genetic algorithms to achieve more complex behaviours. Finally, genetic algorithms and learning techniques are associated in the last experiment, showing a high adaptive structure.

An important effort has been made to show both advantages and disadvantages of each technique, in order to provide the necessary elements for the continuation of this research activity.

Table des matières

À Judith et Luca

Introduction

Dans le cadre de la robotique, la *robotique mobile* joue un rôle à part. Contrairement aux robots industriels manipulateurs qui travaillent de façon autonome dans un grand nombre d'usines automatisées, les robots mobiles sont très peu répandus. Cette situation n'est certainement pas due au manque d'applications possibles. Dés qu'on dispose de la mobilité, on peut imaginer des robots facteurs, nettoyeurs, gardiens, démineurs, explorateurs, jardiniers et beaucoup d'autres. La faible diffusion est surtout due au fait que ces tâches ont une complexité bien supérieure à celles effectuées par des robots manipulateurs industriels. Le monde dans lequel un robot mobile doit se déplacer est souvent très vaste, partiellement ou totalement inconnu, difficilement caractérisable géométriquement et ayant une dynamique propre.

Les méthodes de conception testées jusqu'au milieu des années 80 dans le domaine de la robotique mobile se sont souvent appuyées sur celles de la robotique industrielle des bras manipulateurs. Or ces méthodes impliquent une très bonne connaissance du problème, qui se traduit dans la modélisation du système en question. C'est sur la base de cette modélisation que l'ingénieur opère pour trouver une solution. Malheureusement, la modélisation de systèmes complexes comme les robots mobiles, n'est possible que dans des cas très simples, ce qui limite beaucoup leur application. A partir de cette constatation, plusieurs nouvelles approches ont été développées et testées.

Parmi ces méthodes, un grand espoir a été mis dans les réseaux de neurones artificiels, copie informatique des cellules qui forment le cerveau animal. Malheureusement, le fonctionnement de ce type de structure très efficace n'est pas totalement maîtrisé. Il est ainsi très difficile, à partir d'un problème donné, de définir quel réseau de neurones pourra le résoudre. Malgré cela, les réseaux de neurones ont été utilisés avec succès dans de nombreux problèmes, comme par exemple la conduite visuelle et automatique d'une voiture [Pomerleau93].

Né seulement à la fin des années 80, le domaine de la *vie artificielle* (*Artificial Life*) se propose de reproduire, au niveau informatique, la naissance de la vie. Parmi les méthodes utilisées dans cette communauté scientifique, on trouve les algorithmes génétiques. Cette méthode d'inspiration biologique a été mise au point par des ingénieurs [Rechenberg73][Holland75] pour l'optimisation de solutions à des problèmes complexes. Elle s'inspire du concept de l'évolution naturelle (darwinisme) et a prouvé son efficacité dans de nombreuses simulations.

Au début des années 90, la technique des algorithmes génétiques a commencé à être appliquée au contrôle de robots mobiles. Les résultats ont été très encourageants, donnant lieu à un domaine appelé *robotique évolutive* (*evolutionary robotics*). Ce travail se situe dans ce contexte, mais vise à mieux comprendre les possibilités des réseaux de neurones et des algorithmes génétiques appliqués à des robots réels. Dans ce but, des outils de travail basés sur les robots réels ont été développés. Six expériences ont été conçues afin d'analyser les avantages et inconvénients de ces approches dans l'optique d'une application concrète. Malgré cela, cette analyse reste une étude de base qui vise à mieux orienter des nouvelles recherche plutôt que permettre des applications industrielles.

1 CADRE ET BUT DE L'ÉTUDE

Ce chapitre a pour but d'introduire le sujet de ce travail en le situant brièvement par rapport à quelques repères. Après une définition de la problématique générale de la robotique mobile autonome (1.1), on va préciser ce qu'on entend sous le terme *autonomie* (1.2) et sa relation avec la complexité (1.3). L'importance de la relation entre le robot et son environnement dans ce cadre particulier est soulignée dans la section suivante (1.4). Pour bien situer l'approche utilisée, on va illustrer la relation très enrichissante qui peut s'instaurer entre biologie et technique (1.5) et les formes que cette collaboration a pris en robotique mobile autonome (1.6). Les deux dernières sections présentent le but de ce travail (1.7) et en résument la structure (1.8).

1.1 Le domaine de la robotique mobile autonome

Dans ces dernières décennies on a assisté à un développement considérable de la robotique. L'outil *robot (ou bras) manipulateur* a permis d'automatiser quelques domaines industriels, avec des exemples très frappants dans l'industrie automobile. Pourtant, malgré une forte croissance dans la première moitié des années 80, le développement de la robotique a subi un net ralentissement dans la seconde moitié des années 80 et n'a repris sa croissance qu'au début des années 90, après une meilleure compréhension des possibilités des robots. Effectivement, ce type de bras manipulateur n'a permis d'automatiser que des tâches simples, répétitives et à grande échelle, qui ne nécessitent aucune prise de décision ou seulement des prises de décision très élémentaires de type booléen ou avec des conditions très claires. Des exemples de ce type de tâches sont la soudure par point, la peinture d'une carrosserie, le remplissage d'emballages ou la pose sur gabarit. Ces opérations sont faites par des robots de façon aveugle, répétitive et toujours identique, avec des conditions d'arrêt au cas où les pièces à manipuler manquent ou une personne s'introduit dans la zone de sécurité. La détection peut se faire par une simple barrière optique, par exemple. Ainsi les robots les plus utilisés actuellement sont programmés de façon très explicite et obéissent strictement à des programmes détaillant toute opération, même la plus élémentaire.

Or, un bien plus grand nombre d'applications nécessite des opérations plus complexes et une meilleure flexibilité, comme par exemple le tri d'objets, le contrôle de qualité, le câblage, la manipulation de pièces en petite quantité et d'autres encore. Dès que la manipulation implique un robot de plus de 4-5 axes (ou degrés de liberté), dès que l'on demande une grande flexibilité (petites séries) ou bien une perception de la pièce à manipuler (orientation, forme, etc.), l'homme est souvent plus rentable que la machine. Dans ce type d'applications les robots industriels n'ont pas une autonomie de travail suffisante pour être rentables, ou même seulement utiles, et ceci à cause du manque de perception et d'intelligence des systèmes de contrôle actuellement dans le commerce.

Dans la branche de la robotique qui s'occupe de systèmes mobiles, cette tendance est encore plus nette. Les applications qui peuvent utiliser des robots mobiles program-

més de façon très explicite et obéissant strictement à des programmes détaillant toute opération sont peu nombreuses. Même dans des cas simples comme le transport de matériel dans une entreprise ou dans des hôpitaux, ou le nettoyage de surfaces pré-déterminées et invariables, le robot est confronté à des prises de décision multiples et complexes, ainsi qu'à des espaces de travail beaucoup plus importants que ceux nécessaires au travail d'un robot manipulateur classique.

En robotique industrielle le concept important est le *positionnement* et le *déplacement* du bras manipulateur, basé sur des capteurs de position internes au robot et parfaitement contrôlés. En robotique mobile il s'agit d'*orientation* et de *navigation* du robot, les deux basées sur une perception du monde extérieur. Ceci implique un système sensoriel bien plus sophistiqué que celui d'un robot manipulateur industriel. De plus, l'environnement qu'un robot mobile doit percevoir est bien plus variable que le posage de travail d'un bras manipulateur. Un robot aspirateur, un robot distributeur de courrier, un robot utilisé pour la récolte de fruits ou pour la recherche de mines, un robot de surveillance, tous doivent prendre des décisions importantes et non triviales dans un environnement géométriquement complexe et très changeant. Ceci oblige ce type de systèmes à avoir une perception de l'environnement performante, complétée par une plus grande capacité de traitement de l'information afin d'atteindre une autonomie de travail intéressante. Ainsi on parle de *robots mobiles autonomes*, capables d'exécuter des travaux sans besoin d'aide humaine. Le but reste très semblable à celui des robots manipulateurs industriels. Le terme autonome sert principalement à différencier ce type de robots mobiles de ceux qui sont télécommandés, et qui ne disposent d'aucune capacité décisionnelle propre.

Ainsi, les robots mobiles autonomes peuvent être considérés comme le but ultime d'une branche qui se situe entre l'intelligence artificielle et la robotique. Ce domaine et ses méthodes sont très différents de celui de l'actuelle *robotique industrielle*, qui s'occupe de robots manipulateurs fixes, utilisés de nos jours dans les chaînes de fabrication. Le premier domaine est caractérisé par les termes *mobile* et *autonome*, qui s'opposent aux caractéristiques *à base fixe* et *explicitement programmé* du deuxième. Ces caractéristiques de base ont des répercussions très importantes sur le type de robot utilisé et surtout sur le type de système de contrôle qui le pilote. Le tableau 1 illustre quelques unes de ces caractéristiques.

Robotique...	... industrielle	... mobile autonome
Environnement de travail	Généralement très bien connu, créé et contrôlé par l'homme	Partiellement ou totalement inconnu, naturel, ayant une dynamique propre
	Géométriquement simple	Complexe
	Clairement mesurable	Mobile, mou, flou
	Limité	Pratiquement infini
Interaction avec le monde	Avec peu de capteurs	Basé sur les capteurs
	Par des opérations simples	Mouvements complexes
	et répétitives	et uniques

Robotique...	... industrielle	... mobile autonome
Structure de contrôle nécessaire	Basé sur des modèles du monde Action réalisée en boucle ouverte, essentiellement sans rétroaction de capteurs	Pas de modèle du monde utilisable Boucle sensori-motrice fermée Réactivité Représentation du monde dynamique Adaptation

Tableau 1: Caractérisation de quelques aspects différentiateur entre *robotique industrielle* et *robotique mobile autonome*.

Toutes ces différences ont un point en commun, à savoir la grande complexité de la robotique mobile autonome face à la robotique industrielle. La différence du degré de complexité entre la robotique mobile autonome et la robotique industrielle est semblable à celle présente entre le monde naturel et le monde artificiel créé et maîtrisé pas l'homme. Elle est énorme et difficilement mesurable. Le monde naturel n'est que très partiellement analysable et encore moins maîtrisé. Créer un robot mobile qui puisse, par exemple, surveiller une forêt et annoncer des situations anormales, c'est introduire un produit de notre technologie dans un monde qui est hors de portée de nos mathématiques, de notre physique et de notre informatique. Afin de réussir dans ce défi, il est nécessaire de modifier nos méthodes de conception pour garantir une cohérence entre le robot et son environnement et pour donner au robot un degré important d'autonomie, qui lui permette de faire face à un monde complexe et en mutation continue, dont nous ne pouvons pas décrire les détails.

1.2 Le concept d'autonomie

Un terme clé du titre de ce travail est *autonome*. Dans la robotique mobile on trouve très souvent l'utilisation de ce terme dans des contextes et à des niveaux très différents. En passant en revue quelques définition, on va essayer de mieux définir l'utilisation du concept d'*autonomie* dans le contexte de ce travail:

- Le niveau d'autonomie le plus élémentaire que l'on rencontre souvent est l'autonomie énergétique. Dans le Grand Robert Electronique figure en effet aussi, sous la définition d'autonomie, "Distance que peut franchir un véhicule, un avion, un navire sans être ravitaillé en carburant", ce qui évoque l'autonomie énergétique d'un ordinateur portable ou d'un robot mobile. Ce type d'autonomie peut aussi être informatique, comme le définit le Grand Robert Electronique: "◊ 4. Inform. Qui n'est pas connecté à un calculateur central; qui est indépendant des autres éléments d'un système". Nous allons appeler ce type d'autonomie énergétique et informatique ***autosuffisance***. Un robot, ayant des accumulateurs embarqués, qui arrive à se recharger automatiquement ou qui a une unité de calcul embarquée est donc autosuffisant mais pas forcément autonome. Il n'est même pas néces-

saire qu'un robot autonome soit autosuffisant, car un robot connecté à un ordinateur central peut être un tout parfaitement autonome du point de vue décisionnel. L'autonomie sur laquelle nous allons nous concentrer concerne uniquement le niveau de la prise de décision du robot.

- L'autonomie a été souvent limitée à des cas précis. Il existe des robots qui transportent des objet à travers les usines en suivant des pistes balisées et réservées. Dans ce cas ils suivent des lois décrites à l'avance et bien déterminées; un événement imprévu déclenche l'arrêt immédiat et demande une intervention humaine. Nous allons définir ces comportements comme ***automatiques***, et non autonomes. L'autonomie permet de faire face à des situations non prévues, en prenant des décisions appropriées.
- L'autonomie a été souvent limitée à des échelles temporelles et spatiales bien précises. Un robot à qui l'on dit d'avancer d'un mètre puis de s'arrêter, car on sait que cette trajectoire est possible, pourrait être défini autonome sur cette distance s'il s'occupe d'aller droit et de ne pas dépasser le mètre. Ce type d'autonomie implique normalement que le robot arrive à s'arrêter devant un obstacle imprévu qui viendrait se mettre sur sa route. Mais en dehors de ce mètre le robot n'est pas autonome, n'ayant même pas pris seul la décision de parcourir le mètre. Cet exemple peut sembler simpliste, mais il démontre bien la discordance dans la perception de l'autonomie. Beaucoup de chercheurs considèrent un robot mobile autonome en tant qu'autonome dans sa mobilité, et pas au niveau de la prise de décision. Selon [Hoppen92] et [Knieriemen91], un robot mobile autonome doit être en mesure de se déplacer d'un point A à un point B de façon autonome mais selon un plan qui lui est donné. Nous allons appeler ce type de système ***véhicule guidé automatiquement*** (en anglais AGV, pour *Automatic Guided Vehicle*), dans le sens que le guidage de l'appareil est autonome, mais pas la décision. Un robot autonome doit donc l'être au minimum pendant toute sa durée de vie et dans toutes ses actions. Il doit être en mesure de prendre des décisions en fonction de la tâche, normalement d'une certaine complexité, qui lui a été donnée.

Entre autres, le Grand Robert Electronique définit l'autonomie comme "Droit, fait de se gouverner par ses propres lois". C'est justement au niveau de la création des lois que l'autonomie devient réelle, comme le résume très bien Steels dans [Steels95]. C'est seulement avec une capacité décisionnelle qui lui permet de créer ses propres lois de contrôle, qu'un système peut faire face à un monde imprévisible, complexe, en mutation continue et méconnu par le concepteur même du système.

Ainsi, le fait d'avoir un système que l'on veut autonome a des répercussions importantes sur le rôle de l'ingénieur. Le fait que le système soit capable de créer ses lois implique une perte importante du contrôle de la part de l'ingénieur. Dans le cadre de ces systèmes qui se rapprochent de plus en plus à des êtres vivants (et ceci est le but déclaré de beaucoup de chercheurs [Cliff94]), le rôle de conception de l'ingénieur doit se transformer pour s'approcher au rôle d'instigateur, de motivateur et d'analyste qu'ont le biologiste et le psychologue.

1.3 Approcher l'autonomie et la complexité

Il y a plusieurs façons d'approcher le problème de la conception d'un robot autonome pouvant travailler dans des environnements complexes. De façon très qualitative, on peut représenter les degrés de complexité de l'environnement et de la tâche ainsi que l'autonomie d'un robot sur un graphique ayant sur ses deux axes ces deux paramètres (voir figure 1).

Si on essaie de situer l'état actuel de la recherche dans ce domaine, on se rend compte que complexité et autonomie sont souvent exclusifs (courbe A-B-C, figure 1). Les robots mobiles utilisés dans des environnements très complexes sont normalement et presque uniquement téléopérés, c'est-à-dire avec une autonomie décisionnelle très limitée, voire nulle (point A de la figure 1) [Bejczy94]. Les robots mobiles industriels parmi les plus autonomes sont les robots travaillant dans des hôpitaux ou des entreprises pour le transport de matériel, les robots de surveillance et les robots de nettoyage. Même dans ces cas, l'autonomie est réduite à des opérations simples, telles que l'exploration aléatoire, le balayage d'une surface pré-déterminée ou le suivi d'un parcours, laissant la plupart des décisions à l'homme [Schofield95][Pellerin95]. Même pour atteindre cette autonomie très restreinte, la complexité de l'environnement doit être limitée considérablement. Ainsi les conditions des parcours des robots doivent être bien décrites par une marque sur ou dans le sol, un système de repérage ou un plan de l'usine. En même temps les parcours doivent être libres, le sol bien plat et les prises d'énergie éventuelles bien reparties. L'autonomie que le robot atteint est donc limitée à des prises de décision simples (point B de la figure 1).

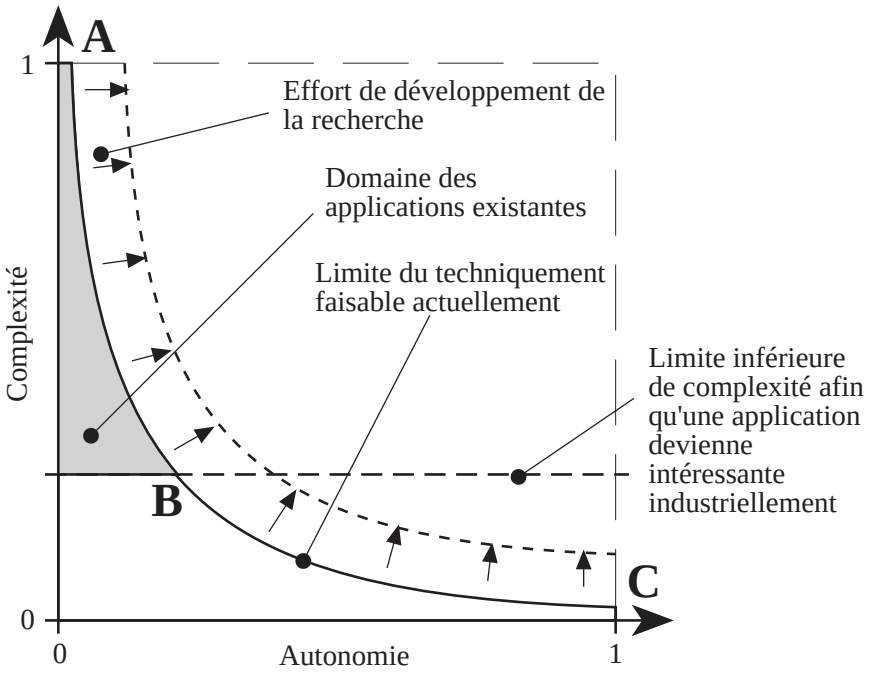


Figure 1: Représentation de la situation des développements actuels par rapport aux deux paramètres de complexité et autonomie

Parmi les application existantes, la tondeuse à gazon solaire et complètement automatique de Husqwarna est probablement un cas exceptionnel. Cet appareil fait face à un jardin réel dans toute sa complexité géographique et météorologique, étant limité seulement par une barrière magnétique qui limite la zone à tondre. Malgré la relative simplicité d'une pelouse, cette tâche reste à ma connaissance une des plus complexes à avoir été réalisée avec un robot mobile. Ceci est confirmé indirectement par le prix du robot (environ 4000.- SFr) qui limite considérablement son introduction sur le marché.

Pour arriver à maîtriser une autonomie plus grande, la complexité doit être tellement réduite, que l'on entre dans le domaine de ce que l'on appelle des *toy-problems*, à savoir des problèmes académiques simplifiés à l'extrême et donc pas intéressants pour des applications réelles. C'est seulement pour des problèmes de ce type, comme l'évitement d'obstacles, par exemple, que l'on arrive à concevoir des systèmes complètement autonomes, qui peuvent prendre des décisions face à presque n'importe quelle situation (point C de la figure 1).

La recherche actuelle qui vise à la création d'un robot mobile autonome fonctionnant dans un environnement complexe (coin supérieur droit du graphe de la figure 1) essaie d'emprunter tout chemin qui va de la courbe actuelle de faisabilité (courbe A-B-C de la figure 1) au coin supérieur droit du graphe. Parmi ce grand nombre de voies possibles on peut reconnaître deux grands types d'approches, à savoir:

- l'approche *télé-opération* qui part d'un robot téléopéré dans un environnement complexe (point A de la figure 1) et essaie de lui ajouter des niveaux d'intelligence et d'autonomie afin de rendre la téléopération plus aisée, se déplaçant ainsi horizontalement sur le graphe vers le coin supérieur droit. Cette approche est suivie, par exemple, par la NASA dans ses programmes d'exploration spatiale [Schilling95]. Ce genre d'application exige l'utilisation d'un robot mobile dans un environnement réel complexe tout en utilisant la technologie actuelle.
- l'approche *robotique autonome* qui ne traite que des robots visant à être autonomes dés le début. On considère ici initialement des tâches très simples (point C de la figure 1) pour évoluer verticalement dans la direction d'une complexité croissante.

L'avantage de la première approche réside dans le fait que l'on travaille dès le début avec l'environnement cible. On peut ainsi mieux comprendre les problèmes, surtout grâce au fait que l'on commence à les résoudre soi-même, en téléopérant le robot. De l'autre côté, cette approche tend à compenser les besoins de l'homme plutôt qu'à créer une vraie autonomie au niveau du robot. Ceci mène à une structuration du système de contrôle du robot par couches, chacune ayant une interface avec les commandes humaines et n'ayant aucune autonomie. On tend donc à traduire ce que l'homme fait, plutôt que comprendre et réaliser ce dont le robot a besoin. Cette tendance conduit souvent à un surdimensionnement du système de contrôle et à la création de faux problèmes (voir aussi le *frame of reference problem*, dans la section 1.6.1).

Quant à la deuxième approche, son avantage est qu'elle se soucie de l'autonomie du système dès le début. Ceci change radicalement la stratégie de développement, car les

besoins pris en compte peuvent être ceux du robot et non ceux de l'opérateur qui commande le robot. La structure de contrôle se trouve donc mieux adaptée aux problèmes rencontrés. L'inconvénient de cette approche est que l'on reste souvent très loin des problèmes réels et des applications visées.

Malgré les avantages de proximité aux applications et la possibilité d'industrialisation rapide de la première approche, la deuxième s'adapte mieux à une étude des systèmes autonomes à long terme. Elle reste en effet plus proche du développement logique des agents et évite des contraintes artificielles posées par la téléopération. Ce travail est basé sur la deuxième approche, tout en regrettant l'éloignement des applications.

1.4 La cohérence du système avec son environnement

La cohérence entre le programme de contrôle, la machine, sa tâche et son environnement est un aspect fondamental pour le bon fonctionnement d'un système. Dans le monde industriel, ce principe a toujours été considéré comme très important, à savoir que les machines sont crées spécifiquement pour une tâche et un environnement. En essayant d'aller à contre-courant, la robotique industrielle a essayé de montrer, au début des années 80, qu'il était possible de réaliser des machines presque universelles, les robots, capables de tout faire et non dédiées à une tâche spécifique. Malheureusement cette prétendue universalité s'est révélée être utopique et le développement de la robotique a connu une baisse importante dans la deuxième moitié des années 80. Actuellement l'industrie robotique est en reprise, mais la flexibilité des robots industriels est utilisée seulement dans des domaines très restreints (peinture de plusieurs types de carrosseries, soudure de points différents, etc.). Dans ces cas, la variation de la tâche exécutée reste faible et l'équipement utilisé supporte le changement tout en garantissant la cohérence du système avec sa tâche et son environnement. Si la différence entre les tâches augmente, l'équipement nécessaire pour supporter le changement, tout en garantissant la cohérence, devient trop compliqué. On serait amenés, par exemple, à utiliser des bras manipulateurs à 6 degrés de libertés pour effectuer des tâches différentes nécessitant chacune deux degrés de liberté. Le prix d'un tel bras serait très élevé, sa programmation complexe et son efficacité réduite. Dans la plupart des cas, on préfère reconcevoir à chaque fois des simples automates dédiés à la tâche précise, car ce type d'équipement est moins chers, bien plus efficace et finalement plus rentable.

En robotique mobile autonome on a connu les mêmes problèmes, mais sous un angle plus informatique. En visant des machines universelles, on a utilisé l'intelligence artificielle, basée sur la logique et la manipulation de symboles, en espérant pouvoir l'appliquer à n'importe quel domaine. Aussi dans ce cas on n'a jamais atteint les niveaux souhaités. Le problème principal a en effet été la cohérence entre les représentations internes et le monde qui voulait être représenté. D'un côté, la représentation du monde dépend fortement de la tâche et de l'environnement dans laquelle elle est utilisée. Si la représentation n'est pas adaptée ou est trop générale, le système devient complexe, lourd et inefficace. D'autre part, il est très difficile de créer une représentation simple, efficace et fiable du monde extérieur à cause de sa complexité et son ambiguïté [Verschure92].

Les premiers résultats significatifs d'une méthode de développement appliquée sur des robot mobiles réels ont été montrés par Brooks [Brooks91], qui a refusé l'approche universelle et symbolique. Il a adopté une approche qui construit le système de contrôle de façon spécifique à la tâche et base son fonctionnement sur l'interaction du robot avec le monde réel. Ses robots ont montré que la tâche peut atteindre un niveau de complexité intéressant (navigation, recherche d'objets, reconnaissance de personnes) sans symbolique interne mais avec des mécanismes simples et bien adaptés au robot, à la tâche à exécuter et à l'environnement. L'intégration optimale du système de contrôle, de la forme du robot et de ses capteurs à la tâche et au monde dans laquelle elle doit être effectuée, rendent faisables des tâches complexes, comme le nettoyage d'une pièce d'appartement [Ulrich96]. Finalement beaucoup de chercheurs dans le domaine de la biologie ont montré comment ce principe est présent en nature: Francescini a montré comment le système visuel relativement simple de la mouche est adapté à sa façon de se déplacer et permet une détection efficace des obstacles [Franceschini91]. Deneubourg a montré comment l'interaction de comportements simples auprès des fourmis peut donner lieu à des résultats complexes de ramassage et de tri d'objets [Deneubourg91].

1.5 L'approche d'analyse et de synthèse

1.5.1 Le concept

Depuis toujours l'homme a analysé la nature et a essayé de s'inspirer des mécanismes naturels pour des réalisations techniques. L'analyse est le domaine privilégié des biologistes, psychologues et d'autres scientifiques qui essaient de comprendre des systèmes qui fonctionnent pour en tirer des concepts, des lois et des mécanismes. La synthèse, par contre, est le domaine des ingénieurs, qui, à partir de concepts, de lois et de mécanismes, créent des systèmes destinés à fonctionner au service de l'homme. Depuis longtemps, ces deux mouvements ont profité de leur complémentarité. D'un côté, les naturalistes ont pu utiliser la technique (mathématiques, physique, chimie, informatique) pour investiguer, mieux comprendre, et parfois aider au fonctionnement de la nature (médecine). D'un autre côté, les ingénieurs se sont souvent inspirés de la nature et des connaissances des naturalistes, avec des niveaux d'inspiration très divers, allant de la structure mécanique au principe abstrait. Au niveau mécanique, le Velcro® est un exemple typique d'inspiration de la nature. En 1948 George DeMaestral, ingénieur suisse, est intrigué par les petites boules végétales qui s'étaient accrochées à ses chaussettes pendant une promenade. En les observant au microscope il découvre une grande quantité de crochets qui s'accrochent à tout tissu laineux. En 1955 il dépose un brevet sur celui qui est aujourd'hui le bien connu système de fixation Velcro®, contraction de mots *velours* et *crochet*. Dans le domaine des mathématiques, et plus particulièrement dans celui de la théorie de la probabilité, la description mathématique du mouvement brownien est un élément essentiel. Or ce mouvement a été découvert par Robert Brown, un botaniste, qui a remarqué ce phénomène en observant des graines de pollen dans l'eau.

Plus proche de nous, l'intelligence artificielle a essayé de copier les résultats d'intelligence animale et humaine en l'implémentant sur des ordinateurs. Dans ce cou-

rant de pensée se trouve, à côté de la tendance cybernétique de McCarthy et Minsky [Minsky75], l'approche psychologique de Newell et Simon, qui voulaient utiliser l'intelligence artificielle comme outil pour mieux comprendre les processus cognitifs [Newell72]. Avec l'avancement des technologies, les biologistes utilisaient en effet la technique non seulement comme outil de travail, mais comme moyen de validation par la simulation. On rencontre la même situation dans le domaine des réseaux de neurones, avec des biologistes et psychologues comme Franceschini [Franceschini91], Edelman [Edelman89] et Braitenberg [Braitenberg84] d'un côté, qui utilisent les réseaux de neurones artificiels comme outil de test et validation de leur théories du vivant, et des ingénieurs comme Widrow, Mead ou Jutten qui utilisent les résultats comme source d'inspiration en informatique [Widrow93][Mead89][Jutten88].

L'approche synthétique de Franceschini, Edelman et Braitenberg a un succès croissant auprès des biologistes, des psychologues, des neuroanatomistes, et crée un lien très important entre la technologie et le monde vivant. Tant que le scientifique du vivant utilisait la technique seulement comme outil, il n'y avait que peu de retour de la biologie à la technique. L'approche synthétique vise à utiliser la technique pour réaliser un artefact du système biologique étudié afin d'en valider le modèle. L'effort nécessaire pour traduire les analyses en modèles et ceux-ci en artefacts, permet non seulement une vraie prise de conscience des problèmes de la part du biologiste, mais crée un transfert de connaissances entre les sciences du vivant et la technique.

La robotique mobile est un domaine très proche du monde animal, car les robots mobiles doivent faire face à des problèmes de mouvement, navigation, saisie, reconnaissance, etc. qui sont très similaires à ceux résolus par les animaux. C'est pour cette raison que l'approche synthétique peut jouer un rôle très important dans ce domaine, en permettant de mettre en évidence des phénomènes très intéressants pour les deux disciplines. La section suivante montre comment l'approche synthétique peut apporter autant au biologiste qu'à l'ingénieur.

1.5.2 Un exemple: la mouche et le robot de Franceschini

La mouche est capable de voler dans des environnements très denses et dynamiques en utilisant sa perception visuelle et sans entrer en collision avec les obstacles. Cet insecte dispose de deux yeux, ayant chacun 24'000 capteurs de lumière regroupés sous 3'000 facettes de 8 capteurs chacune, ce qui paraît presque ridicule face à nos caméras vidéo qui disposent, elles, de l'ordre de grandeur d'un million de pixels bien alignés. Mais contrairement aux pixels des caméras, les facettes de l'oeil de la mouche sont agencés pour concentrer un grand nombre de récepteurs sur l'avant avec une plus faible densité sur les côtés. Dans la partie supérieure de l'oeil, des photorécepteurs spécialisés sont réservés à la poursuite en vol pour l'accouplement. L'information provenant des photorécepteurs est traitée en parallèle et de façon analogique par une structure qui compte environ un million de neurones. Cette structure commande 17 paires de muscles qui ajustent l'amplitude, la fréquence et l'angle d'attaque des ailes. Le positionnement de la tête est lui-même assuré par 21 paires d'actionneurs qui permettent d'utiliser correctement le système visuel.

Nicolas Franceschini étudie depuis longtemps la structure de l'oeil de la mouche [Franceschini72][Franceschini92]. A l'aide d'un microscope de stimulation spécial qui exploite l'optique de la facette comme objectif (diamètre 25 µm, distance focale 50 µm) il a mis en évidence les mécanismes de base qui permettent à la mouche de détecter les obstacles qui se trouvent dans son environnement. La méthode consiste à exciter de façon sélective et avec une séquence temporelle précise des paires de récepteurs qui se trouvent sous une seule facette. Ceci est réalisé à l'aide du microscope, utilisé pour focaliser un rayon lumineux sur les récepteurs. L'activité de quelques neurones bien identifiables est mesurée à l'aide d'une électrode introduite depuis l'arrière de la tête de la mouche. De cette façon, Franceschini a montré qu'il existe des neurones dans le deuxième et troisième ganglion optique qui sont sensibles à des mouvements de figures bien précis dans le champ visuel de l'insecte. Ainsi la mouche est en mesure de détecter des flux optiques, et c'est avec cette mesure associée à sa vitesse propre qu'elle peut estimer la distance d'un objet qui se trouve dans son champ de vision.

Pour vérifier ses théories, Franceschini a opté pour la méthode synthétique et a réalisé un robot dans lequel il a implémenté une ligne monodimensionelle de facettes d'oeil de mouche. Cette réalisation essaie de répliquer de façon assez fidèle les aspects biologiques suivants, observés sur la mouche:

- Le principe général de détection d'obstacles par flux optique: le seul capteur utilisé pour la détection d'obstacles est la vision.
- Les déplacements composés de translations et rotations saccadiques: ce type de déplacement a été observé sur la mouche et est bien cohérent avec le système de détection d'obstacles.
- La transmission et l'élaboration du signal analogique et asynchrone: tout le traitement est exclusivement fait de cette façon, de la détection jusqu'aux actionneurs.
- La distribution non-homogène des facettes: dans cette distribution réside une bonne partie du mécanisme de détection.

Ces contraintes on donné lieu à un robot cylindrique de 10 Kg dont la base fonctionne selon le principe du synchro-drive, à savoir 3 roues motrices orientables. Le système de vision est composé de 118 facettes disposée sur le contour du robot selon la distribution observée dans la mouche. Cette distribution est illustrée dans la figure 2: les directions de perception (rayons) sont distribuées de telle façon qu'un mouvement unitaire d'un objet vers le robot ou vice-versa à une distance fixe moyenne *d* (flèches) représente un même déplacement unitaire au niveau perceptif quelle que soit la direction dans laquelle l'objet se trouve.

Ainsi on obtient une mesure de la distance dépendant uniquement de la vitesse relative de l'objet par rapport au robot ou vice-versa. Les détecteurs de mouvement sont placés derrière les capteurs et connectés par une fibre optique qui recueille et transmet la lumière. Ces détecteurs ont pour fonctionnalité de s'activer lorsqu'un signal se propage d'un capteur à son voisin à une certaine vitesse. Cette fonctionnalité permet de détecter, lors de l'avancement rectiligne du robot, la présence d'un objet contrasté à une certaine distance. Il est à noter que ce système de détection peut fonctionner seulement avec des

mouvements rectilignes du robot. Une fois identifiés l'obstacle et sa direction, le robot change de direction pour l'éviter. Ce comportement, qui peut paraître étrange, a toutefois été observé sur les mouches.

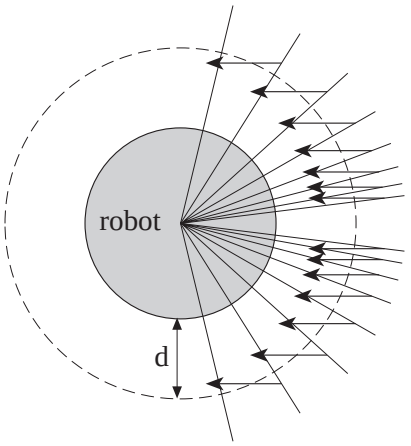


Figure 2: Distribution des directions de perception des facettes de l'oeil, permettant d'obtenir une mesure de la distance dépendante uniquement de la vitesse relative de l'objet par rapport au robot ou vice-versa

Pour motiver le robot et rendre l'expérience plus intéressante, un détecteur de but a été placé au dessus du robot et a été utilisé pour donner une direction préférentielle au robot.

Cette réalisation artificielle a permis à Franceschini de montrer que les principes de fonctionnement de l'oeil de la mouche qu'il avait présenté étaient vraisemblables. Ce résultat, intéressant en soi, en cache d'autres, qui mettent bien en évidence l'interaction entre biologie et robotique. Par exemple, lors de la conception du robot, la précision de positionnement des capteurs dans les facettes nécessaire au bon fonctionnement du système s'est révélée être très grande, à un point de paraître invraisemblable biologiquement. Les observations ultérieures faites sur la mouche ont permis de découvrir des muscles qui ajustaient très précisément les photorécepteurs derrière les facettes de l'oeil, afin d'atteindre cette précision.

Ce dernier exemple de passage entre l'analyse biologique, la synthèse robotique et à nouveau l'analyse biologique est exemplaire de l'apport que l'approche synthétique peut donner au biologiste à la découverte du monde animal. Cette méthodologie de travail permet, en effet, de mettre à dure épreuve les modèles dans leur intégrité.

1.5.3 Discussion

Du point de vue de l'ingénieur et de la robotique, il est possible de tirer plusieurs enseignements du travail de Franceschini:

- Dans le monde animal, il existe des mécanismes relativement simples et bien adaptés pour faire des tâches considérées compliquées dans le monde de l'ingénieur. L'évitement d'obstacles par vision passive (sans émission active de lumière par le robot) est un exemple frappant de ce fait. Avec

Franceschini, on se trouve devant un robot qui fait de la vision en temps réel (sans même utiliser un processeur), alors que ce même problème est considéré comme étant très compliqué par les professionnels de la vision artificielle qui travaillent avec des machines et des algorithmes sophistiqués.

- Les représentations du monde faites à l'intérieur d'un système de contrôle naturel sont parfois très différentes de ce que l'on peut s'imaginer. Elles sont adaptées à la tâche que le système doit exécuter, mais normalement très différentes des concepts que nous pourrions avoir de ces représentations. La mouche ne doit pas distinguer le type d'obstacles pour l'éviter, ainsi qu'elle n'a pas une représentation cartésienne de l'environnement.
- L'adaptation du mécanisme à la tâche que le robot doit exécuter est très importante. L'évolution a très finement façonné les mécanismes de vision de la mouche à son mode de survie et à son environnement. La partie de l'oeil pour la détection d'obstacles en est un exemple, la différentiation de la partie supérieure de l'oeil pour le suivi en vol pour l'accouplement en est un autre.
- Il est très important d'optimiser le mécanisme dans son ensemble. Ainsi le principe de fonctionnement, le type et la quantité de capteurs, le type de traitement et le type d'actionneurs doivent être considérés dans leur ensemble et être conçus comme un tout. La bonne répartition des fonctionnalités aide à simplifier remarquablement le système. La simple répartition géométrique des capteurs, par exemple, facilite sensiblement le traitement qui se fait ensuite. En plus, le type de déplacements (rectilignes) rend le mécanisme de détection moins complexe.
- Le parallélisme, fortement utilisé par la nature à différents niveaux (neurones, cellules, fonctionnement des plantes, populations d'individus) apporte des éléments intéressants comme la puissance de calcul élevée, un temps de réaction amélioré en conséquence, la robustesse, la possibilité d'utilisation d'un grand nombre d'éléments simples et l'obtention d'un résultat complexe.

1.6 Les méthodes de conception de robots autonomes

Les méthodes de conception sont généralement regroupées dans deux grandes catégories, à savoir top-down et bottom-up.

La première classe de méthodes (top-down) est la plus classique du monde de l'ingénieur. On commence par définir la tâche finale à exécuter. Ensuite on la subdivise en sous-tâches nécessaires à son accomplissement. Ces sous-tâches sont, à leur tour, subdivisées de façon itérative jusqu'à l'obtention de tâches élémentaires réalisables. Une fois que celles-ci sont réalisées, on reconstruit le système complet en se basant sur la décomposition faite préalablement. Cette approche, très utilisée dans les développements techniques, nécessite des conditions très fortes: Il faut avoir une très bonne maîtrise du problème afin de le subdiviser correctement. La subdivision doit en effet générer des

sous-tâches bien complémentaires et atteindre les tâches élémentaires réalisables. Si cette technique a fait ses preuves dans le monde industriel actuel, elle est très difficilement applicable dans le domaine de la robotique mobile. Comme le problème est inconnu sous plusieurs aspects liés au monde réel et à son interaction avec le robot, il n'est pas facile de définir, à partir du résultat que l'on veut obtenir sur un robot mobile autonome, quelles sont les structures nécessaires et les mécanismes à utiliser pour réaliser ce résultat. La difficulté est similaire à celle rencontrée lorsque l'on essaie de comprendre les structures et le mécanismes en jeu dans le comportement d'un animal à partir de l'observation du comportement obtenu. Ainsi cette approche est de plus en plus rejetée par un grand nombre de chercheurs du domaine de la robotique mobile autonome.

La deuxième approche (*bottom-up*) propose de partir de tâches simples pour ensuite les complexifier jusqu'à atteindre un niveau intéressant. Cette approche a l'avantage de construire sur des bases solides, établies préalablement, mais pose aussi de sérieux problèmes. Ainsi la construction du système ne peut pas être purement *bottom-up*, car il est nécessaire de diriger le travail vers un but. Or, une approche *bottom-up* par excellence comme la *subsumption* [Brooks86], par exemple, ne donne pas d'indication sur comment diriger la construction. Il en résulte des problèmes de structure qui, couche après couche, donnent lieu à un système inextricable, dans lequel toute addition d'un nouvel élément nécessite la modification du système construit préalablement. L'avantage de cette méthode, pourtant, réside dans le fait qu'elle se base sur des sous-tâches réalistes et permet de détecter les problèmes éventuels de façon rapide et systématique.

Afin d'adopter une méthodologie efficace, il est nécessaire de se situer entre ces deux extrêmes, c'est-à-dire construire sur des bases solides et analyser les problèmes de façon systématique et constructive tout en ayant une claire direction de développement qui dirige l'effort vers un but pré-défini.

A l'intérieur de ces deux grandes classes de méthodes, on peut distinguer plusieurs méthodes particulièrement connues, qui diffèrent par les motivations, les principes et les implémentations utilisées. Toutes essaient de créer des systèmes cognitifs, capables de gérer de l'information de façon intelligente et autonome, afin, par exemple, de contrôler un système physique tel qu'un robot. Comme on le verra par la suite, la plupart de ces approches ont pris comme source d'inspiration la biologie. L'animal, tel que l'insecte ou l'homme, est un très bon exemple de système autonome. La différence entre les méthodes réside dans le niveau d'inspiration, et dans ce que l'on considère pertinent au problème posé: le comportement pour les uns, la structure de traitement des données ou les méthodes de traitements pour d'autres.

Quelques-unes des ces approches sont présentées dans les sections suivantes.

1.6.1 L'intelligence artificielle

L'intelligence artificielle (en sigle IA) a généré les premiers essais de simulation et de développement de systèmes cognitifs liés à la robotique [Chatila94]. Les mécanismes de la nature qui ont été pris comme base de travail sont de très haut niveau, tels que la logique de raisonnement, le langage, la planification, les concepts, les représentations, la mémoire et d'autres. Pour cette ligne de pensée, l'intelligence est conçue comme la capa-

cité de manipuler une série de symboles à travers des règles logiques. On se base donc sur une représentation symbolique du monde que l'on manipule en utilisant la logique formelle. En conséquence, le monde réel est conçu comme mathématiquement modélisable et donc clairement divisible en entités associables à des symboles.

Cette approche, qui est associée à des méthodes de conception *top-down*, a donné des preuves brillantes de fonctionnement dans des problèmes logiques abstraits, mais rencontre des problèmes sérieux en robotique mobile autonome. Le problème principal dérive du choix de la symbolique utilisée, de sa définition et de l'association entre symboles et objets du monde réel. Comme mentionné plus haut, le choix de la symbolique dépend de la tâche à exécuter, et il est ainsi très difficile de trouver une symbolique qui soit à la fois universelle et efficace. Une fois que le choix est fait et en admettant qu'il est bien fait, la définition même du symbole reste un grand problème: il est impossible de définir objectivement et de façon unique un symbole. Ce problème est connu sous le nom de *symbol grounding problem* [Harnad90]. Enfin, si, dans le modèle, les symboles et les règles logiques, qui en gèrent les relations, sont clairement définis, dans le monde réel, tout est flou, difficilement mesurable et en changement continu. Ainsi il est pratiquement impossible de créer et maintenir un lien objet-symbole en classifiant de façon claire les objets, tout en gardant cette classification à jour (*frame problem*) [Pylyshyn87]. Enfin, dans les systèmes mis en oeuvre selon cette approche, l'expérience de l'agent n'est que très peu prise en compte lors de nouvelles situations. Ce problème est communément appelé *situatedness problem* [Suchman87].

Ces trois grands problèmes de l'intelligence artificielle sont essentiellement dus à une conception abstraite de l'intelligence, détachée du monde physique. L'interaction entre le système et l'environnement dans lequel il agit a été cachée ou simplifiée en se penchant uniquement sur des problèmes de raisonnement. Ainsi des gros problèmes sont survenus quand on a essayé d'interfacer cette intelligence avec le monde physique. C'est à ce moment qu'on a découvert qu'il était difficile, par exemple, de reconnaître une chaise, alors que le système logique se basait sur ce concept. Cette interaction agent-environnement, comme mentionné plus haut, joue un rôle essentiel dans le fonctionnement du système. L'échec de l'approche *intelligence artificielle* en est une conséquence.

Un dernier problème est celui du référentiel utilisé pour la conception des systèmes artificiels connu sous le terme de "*frame of reference problem*". En intelligence artificielle, le modèle est l'homme et sa façon abstraite de traiter les problèmes. Or, ce modèle ne s'applique qu'aux problèmes auquel l'homme est confronté. Si le robot mobile doit faire face à des problèmes beaucoup plus simples, du niveau, par exemple, de ceux rencontrés par une fourmi, l'application de méthodes de résolution basées sur l'intelligence humaine est inadéquate et ne respecte pas la contrainte de cohérence entre le système de contrôle, le robot, la tâche et l'environnement. La nature a d'ailleurs doté la fourmi de capacités et méthodes de résolution des problèmes qui sont différentes de celles de l'homme.

Enfin, l'intelligence artificielle classique ne respecte pas l'autonomie de choix de décision du système, mais elle impose des règles externes et, en bonne partie, fixes. Malheureusement, ce problème est commun à beaucoup d'autres approches.

1.6.2 La “nouvelle IA”

Portée par Brooks [Brooks86], la *nouvelle IA* essaie d’éliminer certains des problèmes rencontrés par l’intelligence artificielle classique. Les aspects symbolique et *top-down* sont remplacés par une approche basée sur l’interaction avec l’environnement et fondamentalement *bottom-up*. Brooks aime répéter que “le monde est la meilleure représentation de lui-même” [Brooks91b] et que la représentation interne, vu qu’elle ne peut être que limitée, est inutile. Une grande importance est donc donnée aux capteurs et actionneurs, ainsi qu’au traitement pertinent des informations. De plus la structure du système de contrôle est revue: au lieu d’avoir des modules fonctionnels, Brooks subdivise le système de contrôle en modules comportementaux. Les modules fonctionnent de façon parallèle et sont organisés dans un réseau, appelée *subsumption architecture*, qui fixe les priorités des différents modules. Cette structure permet, comme le veut l’approche *bottom-up*, de complexifier le comportement par l’adjonction de nouveaux modules au réseau.

Brooks et sa théorie ont eu le grand mérite de s’attaquer aux problèmes de base de l’IA classique en montrant des résultats sur des robots réels et des tâches non triviales. Il a implémenté des structures parallèles, en améliorant la robustesse des systèmes. Le fonctionnement de ses robots a fait de lui l’une des personnes les plus connues dans le domaine de la robotique mobile. Malheureusement, le problème principal auquel il s’attaquait, à savoir les symboles de l’IA, a été critiqué mais pas résolu: on reproche à Brooks de s’être limité à des systèmes réactifs, c’est-à-dire sans mémoire, sans représentations internes, dont le fonctionnement se base uniquement sur les entrées sensorielles instantanées. Tout en allant plus loin que ce que l’on s’attendait, il n’a jamais réussi à réaliser des tâches réellement complexes. De plus, la conception des modules de base de sa structure a été faite de façon classique, avec des problèmes similaires à ceux rencontrés auparavant. Enfin, la possibilité de complexifier à volonté le réseau est purement théorique, car elle dépend entièrement du concepteur et de la façon dont celui-ci crée le réseau. En effet, pour réussir, cette approche ne peut pas seulement se baser sur une croissance *bottom-up* pure, mais nécessite une planification *top-down* de l’extension du réseau.

Brooks, après une période de travail peu fructueuse sur son dernier robot, le très complexe Attila, a actuellement abandonné presque totalement les robots du niveau insecte pour passer à des robots humanoïdes [Brooks93] et à des projets d’interfaces multimedia [Brooks96]. Malgré ce changement radical, il n’a jamais tiré de conclusions critiques sur son travail, ni donné des raisons valables de son changement, ce qui laisse quelques doutes sur la réelle efficacité de sa méthode.

1.6.3 Le connectionisme

Le connectionisme est né comme une tentative de s’approcher des mécanismes biologiques de la cognition. L’attention n’est plus donnée aux fonctionnalités de raisonnement (comme en IA classique) ou au comportement de l’individu (nouvelle IA) mais à la microstructure du système nerveux animal. On vise donc à réaliser des structures de traitement de l’information massivement parallèles, appelés *réseaux de neurones artifi-*

ciels (RNA). Les caractéristiques principales de cette approche sont l'apprentissage et le stockage de l'information de façon répartie sur toutes les unités de calcul qui composent le système. Cet apprentissage permet de recueillir les informations sur la base d'exemples et permet ainsi au système de s'adapter à l'environnement dans lequel il est plongé. Les réponses du système peuvent profiter des capacités de généralisation des RNA, dues au type de structure de stockage et de calcul. La structure massivement parallèle permet une meilleure résistance aux pannes et une plus grande rapidité d'exécution si elle est implémentée sur une machine de calcul adéquate. Enfin le type de structure de calcul permet d'aproximer un grand nombre de fonctions non-linéaires. Une description plus détaillée des techniques utilisées est donnée dans la section 4 *L'apprentissage*.

Si cette approche permet théoriquement, grâce à l'apprentissage, d'éviter la phase de modélisation classique du monde que l'on trouve dans l'IA, elle reste toutefois limitée. Les modèles internes, même s'il sont construits par l'apprentissage, sont souvent créés dans des structures et selon des modalités déterminées à l'avance par le concepteur. A ce problème, il faut ajouter la difficulté de concevoir la structure d'un réseau de neurones artificiel, la complexité de son fonctionnement, le choix des règles et des paramètres d'apprentissage, etc. Les réseaux de neurones sont en effet très difficilement analysables, donc difficiles à maîtriser et corriger, au contraire des systèmes de l'IA classique où tout est parfaitement logique et bien défini selon une convention bien connue du concepteur. De plus, trop souvent, les modèles de réseaux de neurones proposés sont utilisés dans deux phases distinctes, l'une d'apprentissage, l'autre d'utilisation, ce qui n'est pas applicable à un système qui doit s'adapter de façon continue. Enfin, la plupart des algorithmes proposés ont des temps de convergence et des modalités d'apprentissage qui ne s'adaptent pas aux contraintes du monde réel.

1.6.4 Les tendances nouvelles

L'échec de l'IA en robotique mobile autonome et les succès relatifs de Brooks restent des points de repère pour la recherche actuelle. La nouvelle tendance est le constructivisme qui propose, comme l'a fait Brooks, une approche *bottom-up* avec une intelligence émergente de l'interaction entre les éléments composants le robot et son environnement, sans utiliser nécessairement des représentations symboliques de haut niveau [Stewart94]. Ces principes permettent de mieux s'approcher du problème de l'autonomie, car ils supposent une auto-gestion du système. Malheureusement, trop souvent, les structures mises en jeu sont très strictement définies et limitent l'émergence à des aspects secondaires, comme par exemple la création d'un certain type de connaissances [Gaussier94].

Le mouvement de l'"Artificial Life" [Langton88] essaie d'utiliser les principes constructivistes associés à la méthode de synthèse afin de comprendre le processus qui génère la vie. La méthode synthétique est justifiée par le fait que les techniques analytiques actuelles ne permettent pas de comprendre les transformations non-linéaires, les propriétés chaotiques et d'autres phénomènes qui caractérisent le vivant. Dans ce mouvement, on laisse donc un peu plus de place à l'autonomie et à l'émergence, en utilisant des techniques évolutionnistes. Ces travaux sont essentiellement basés sur des simulations inspirées des domaines les plus divers, allant de la biologie moléculaire à la morphoge-

nèse des plantes, jusqu'aux comportements intelligents d'organismes ayant un système nerveux. Malheureusement, cette recherche se limite essentiellement à des simulations, sans se pencher attentivement sur la complexité du monde réel.

En robotique mobile autonome, l'approche constructiviste semble très prometteuse, à condition de vraiment laisser la connaissance du système se former sur l'interaction entre le système de contrôle, l'agent physique et l'environnement dans lequel il agit. Le système de contrôle doit être considéré comme un tout avec le système physique qu'il gère et doit fonder son fonctionnement sur des mécanismes d'auto-organisation. La motivation générale de développement doit être la survie, et c'est le problème de l'ingénieur de considérer le développement de l'agent sous ce point de vue.

Ainsi la nature reste un exemple parfait de fonctionnement, mais les mécanismes pris en compte deviennent de plus en plus nombreux et jouent des rôles complémentaires. C'est en considérant cet ensemble cohérent (autonomie, émergence, réseaux de neurones en tant que structure de calcul parallèle et robuste, interaction avec l'environnement, évolution) qu'il semble être possible de réaliser de vrais robots mobiles autonomes.

1.7 Le but de ce travail

En se basant sur les observations faites dans ce chapitre, les points suivants semblent être primordiaux dans la conception de la structure de contrôle d'un robot autonome:

- La complexité du problème:

  L'environnement d'un robot mobile ainsi que le genre de tâches qu'il est tenu de réaliser sont d'une complexité qui n'a jamais été maîtrisée, qui est impossible à modéliser, et qui dépasse en grande partie les méthodes de conception classiques de l'ingénieur, car ces dernières sont basées sur l'analyse, la modélisation et la résolution du problème par un algorithme claire et linéaire.
- L'interaction avec l'environnement:

  C'est à ce niveau que le fonctionnement du robot se réalise. C'est dans cette relation avec l'environnement qu'il est possible de faire face au problème de la complexité, permettant au robot de tester, vérifier, agir et percevoir de manière continue. Une adaptation basée sur cette interaction est une bonne base de fonctionnement pour un robot mobile autonome.
- Le besoin de structures de contrôle simples, mais plastiques et bien adaptées:

  Même des comportements qui apparaissent complexes peuvent être générés par des structures simples et bien adaptés à la tâche [Braitenberg84].

Le but de ce travail est de mettre en évidence les possibilités d'exploitation de l'interaction du robot avec l'environnement, afin de créer une structure de contrôle efficace, qui peut faire face à un certain niveau de complexité grâce à un niveau d'autonomie

important (dans le sens décrit dans la section 1.2). Afin de se limiter à un type de structure, les réseaux de neurones ont étés choisis pour leur flexibilité, leur propriétés de calcul non-linéaire, et le niveau intéressant de l'inspiration biologique.

1.8 L'organisation de ce travail et la structure du rapport

Ce travail est composé de six expériences. Chaque expérience essaie de mettre en évidence un aspect de la relation entre la conception du système de contrôle du robot et l'interaction entre le robot et son environnement réel.

L'ordre de présentation des expériences dans ce rapport ne correspond pas à l'ordre chronologique de réalisation. Des expériences de vérification de concepts de base, présentées ici au début, ont été réalisées après l'observation du concept lors d'expériences plus complexes. Ainsi, l'ordre est donné en fonction de la complexité et de l'autonomie croissante des structures de contrôle mises en oeuvre. La figure 3 montre les domaines explorés dans les six expériences réalisées dans ce travail.

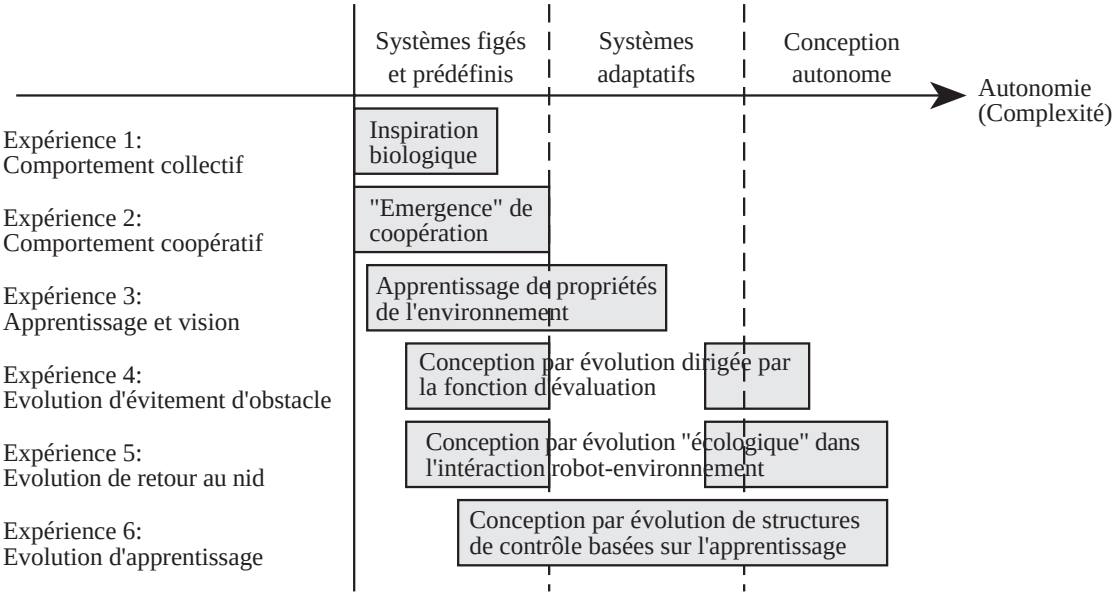


Figure 3: Domaines d'exploration des six expériences

Les premières deux expériences mettent en évidence l'importance de l'interaction robot-environnement, le robot ayant des comportements très simples et programmés de façon classique. On constate l'importance de l'environnement et de l'observation du comportement de la part de l'expérimentateur. De plus, les résultats obtenus mettent en évidence des aspects intéressants du parallélisme, utilisés ici au niveau d'individus mais exploités aussi au niveau des réseaux de neurones.

La troisième expérience permet d'étudier un mécanisme d'adaptation qui commence à exploiter la relation robot-environnement, à savoir l'apprentissage par condi-

tionnement classique en utilisant des réseaux de neurones. Ce type d'adaptation, dans ses formes les plus connues, est toutefois relativement limité et concentré sur des aspects particuliers, définis à l'avance par le concepteur.

La quatrième expérience est un premier essai d'utilisation des méthodes évolutionnistes pour la conception de la structure de contrôle du robot. Cette méthode est également appliquée à des réseaux de neurones et permet une meilleure et plus profonde exploitation de l'interaction robot-environnement. Cet essai met en évidence des aspects prometteurs ainsi que des problèmes propres à cette approche.

La cinquième expérience, très similaire dans sa forme à la précédente, développe l'approche évolutionniste, en exploitant encore plus radicalement l'interaction robot-environnement. Elle fait ressortir toutefois une limitation majeure, à savoir le temps nécessaire au développement.

La dernière expérience combine les adaptations de l'apprentissage et de l'évolution. Les résultats très intéressants exploitent l'interaction robot-environnement à plusieurs niveaux.

La description de ces expériences est précédée par une introduction sur les outils utilisés, qui ont joué un rôle essentiel dans ce travail. De plus, une brève introduction aux réseaux de neurones est donnée avant la description de la troisième expérience.

2 L'ENVIRONNEMENT DE TRAVAIL

Afin de pouvoir mettre en évidence des caractéristiques intéressantes en robotique mobile, il est nécessaire de disposer d'outils d'expérimentation efficaces. Comme en biologie il s'agit non seulement de disposer des sujets d'expérimentation, mais encore des outils qui permettent de recueillir et analyser les résultats. Une grande partie du travail de Franceschini, par exemple, a été de mettre au point le microscope de stimulation qui lui a permis de réaliser les mesures sur l'oeil de la mouche [Franceschini92]. Dans notre cas aussi il s'agit de s'équiper pour pouvoir expérimenter des nouvelles structures de contrôle pour robots mobiles autonomes. Ce chapitre est donc consacré aux outils utilisés couramment et à ceux développés spécialement pour cette recherche.

2.1 Les robots simulés et les robots réels

Quand il s'agit de faire une étude sur le fonctionnement d'un système de contrôle de robots mobiles, beaucoup de chercheurs optent pour une modélisation du problème (robot et environnement) afin d'analyser le problème par pur calcul sur ordinateur. On parle de simulation du problème, simulateur de robot, robot simulé et parfois abusivement de robot tout simplement. Le succès de cette approche est due à des avantages bien connus mais discutables:

1 Ecrire un programme est plus facile et nécessite moins de connaissances que de réaliser un robot physique. Mis à part le domaine de la robotique collective qui implique des dizaines de robots, le gain économique que fait le programmateur en écrivant son simulateur par rapport à acheter un robot du marché est très discutable, si les heures de programmation sont prises en compte. Malheureusement ce dernier point est souvent négligé dans le domaine universitaire. Beaucoup de personnes ont l'impression (souvent fausse) qu'écrire le simulateur revient meilleur marché et se fait plus rapidement que d'acheter un robot et apprendre à le programmer.

   La solution qui consiste à utiliser un simulateur existant permet de limiter les investissements. Souvent, toutefois, le chercheur qui emprunte cette voie reste déçu par les possibilités offertes par le simulateur. Il se retrouve ainsi avec un logiciel dont il n'a pas le contrôle et qui ne satisfait pas ses besoins spécifiques.

2 Dans une simulation on peut accélérer le temps d'exécution. Avec les puissances de calcul actuelles, on peut effectivement élaborer beaucoup de données, mais la vitesse de la simulation va dépendre énormément de ce qui est simulé. Une simulation soignée peut prendre un temps de calcul considérable, si la modélisation de l'environnement et des capteurs est très fine.

3 Dans une simulation on peut faire varier chaque paramètre de l'environnement comme on le désire. Il est effectivement important, lors de certai-

nes expériences, de pouvoir isoler des problèmes comme la friction des roues ou la masse du robot afin de mieux analyser leur influence. Malgré ce fait, changer les paramètres signifie aussi de savoir déterminer les paramètres corrects pour que la simulation soit réaliste.

4 Dans une simulation on peut faire exécuter au robot ce que l'on veut. Il est possible de déplacer un robot d'un coté à l'autre de la pièce par un simple geste sur la souris, ou de le replacer toujours au même endroit pour le test répétitif d'un comportement, ou encore de le faire traverser les obstacles... Cette flexibilité doit toutefois être utilisée avec prudence, en respectant des paramètres réalistes pour le comportement normal du robot, ce qui n'est pas toujours facile à mettre en oeuvre.

5 Dans une simulation on peut modifier les données anatomiques et fonctionnelles du robot. Ceci permet, par exemple, d'améliorer très simplement et rapidement la dynamique d'un capteur, la résolution d'une caméra etc. Ici aussi il faut toutefois définir des paramètres réalistes et les modifier en respectant les contraintes réelles.

6 Dans une simulation sur ordinateur on peut visualiser tout ce qui se passe. L'interface utilisateur de l'ordinateur permet une interaction optimale avec l'expérience. Si cet avantage est réel dans beaucoup de cas, il perd son importance si l'on considère les possibilités de visualisation qui se développent en robotique [Mondada93].

7 Un logiciel de simulation peut être distribué, les résultats répétés, vérifiés et comparés dans d'autres laboratoires. Un robot et son environnement se laissent beaucoup moins facilement reproduire.

Tous ces avantages, même s'ils sont parfois très relatifs, sont contrebalancés par quelques désavantages mineurs et par un problème fondamental: Est-ce que la simulation représente bien le monde réel, au point que les résultats obtenus peuvent être reportés dans la réalité? Il faut bien voir que c'est à cette condition seulement qu'une simulation peut être considérée comme utile.

Comme cela a été illustré dans le chapitre 1 *Cadre et but de l'étude*, les aspects à prendre en considération très attentivement dans le domaine de la robotique mobile sont la complexité et la relation entre système de contrôle, physique du robot et environnement. C'est sur ces deux points que se situe une grande partie du problème. Il est donc fondamental d'en tenir compte.

La complexité de l'environnement porte sur toutes les caractéristiques qui rendent difficile sa détection, qui varient au cours du temps de façon imprévisible, qui influencent le fonctionnement du robot, qui sont couvertes d'erreurs d'estimation, de bruit qui n'est jamais blanc, etc. C'est aussi cette complexité qui ne permet pas la représentation symbolique de l'IA et qui rend pratiquement impossible toute autre modélisation du monde qui pourrait être utilisée dans la commande d'un robot mobile. Essayer de comprendre ce problème en simulation, donc dans un modèle du monde, est simplement contradictoire. La modélisation faite pour la simulation enlève justement l'aspect de la complexité qui est le thème central de ces études.

Pour que l'étude visée dans ce travail ait un sens, il est donc nécessaire de réaliser les expériences sur un robot mobile réel et non pas simulé. La simulation, si elle est faite soigneusement, peut être considérée comme une approche grossière de la réalité et utilisée pour dégrossir les données du problème, en permettant d'avancer rapidement grâce à la rapidité d'évaluation qui est propre de cette approche. Il ne faut toutefois jamais rester au stade de la simulation: la seule vraie vérification doit se faire sur un robot réel.

On peut finalement distinguer plusieurs phases de développements: recherche, pré-étude de faisabilité, et développement de l'application. Nous venons de voir que la première phase de la recherche peut être menée en simulation pure sur ordinateur. Prolonger trop longtemps l'utilisation de la simulation risque de rendre l'étude simpliste, mais le fait d'utiliser trop tôt un robot industriel de grosse taille pour la recherche ou la pré-étude est risqué, car on ne connaît pas suffisamment le problème et l'étude serait lourde et coûteuse. Entre ces deux différentes étapes il faut donc un changement graduel, basé sur des outils efficaces mais pas trop simplistes. C'est dans ce cadre que se situe ce travail ainsi que l'analyse des outils de la prochaine section. Cette dernière a débouché sur le développement du robot Khepera, présenté ensuite.

2.2 Les outils existant (à l'exclusion de Khepera)

Comme mentionné dans la section précédente, la simulation est souvent choisie comme méthode d'analyse à cause des nombreux avantages qu'elle présente par rapport à des expériences basées sur des robots réels. Si on analyse les outils d'expérimentation les plus utilisés en robotique mobile autonome on peut en effet observer qu'ils manquent de flexibilité, de maniabilité et d'efficacité. Le robot est souvent vu comme une machine à microprocesseur dans laquelle on fait exécuter du code et de laquelle on attend un comportement déterminé. Au delà de quelques outils de déverminage, des vrais outils d'analyse du comportement du robot, comme on les trouve dans la biologie, manquent dans la plupart des cas. Ce manque d'outils est souvent remplacé par des outils de simulation ou de *conception assistée par ordinateur*, qui permettent de réaliser l'application sur simulation avant de la transposer sur le robot. Les sections qui suivent se proposent d'illustrer les principes de quelques robots expérimentaux et commerciaux parmi les plus connus ou répandus. Cette liste n'est de loin pas exhaustive. Elle donne toutefois une idée des équipements disponibles et utilisés dans ce domaine.

2.2.1 Les équipements expérimentaux

Par équipement expérimentaux on entend tout système qui est utilisé pour la recherche en robotique mobile réelle mais qui n'est pas disponible commercialement.

2.2.1.1 Le yamabico

Ce robot est probablement un des plus connus au Japon. Il a été développé à Tsukuba, par l'équipe du professeur Yuta [Yuta91]. Il possède un système de locomotion basé sur deux roues motrices latérales et d'une roue folle pour l'équilibre. Sa forme est plutôt verticale, car la structure s'élève en hauteur (entre un mètre et un mètre et demi

selon les configurations) sur une base relativement étroite, comme le montre la figure 4.

Son système sensoriel est basé sur des capteurs à ultrason (qui lui ont donné le nom), mais sa configuration de base comporte aussi des capteurs optiques.

Le corps central est composé d'un fond de panier sur lequel sont connectée des cartes processeurs dédiées aux différents modules sensoriels ou moteurs. Une des cartes est considérée comme maître, et s'occupe de la gestion de haut niveau du robot. Le système opératif de cette carte est propre au robot (MOSRA) et permet une gestion des applications temps-réel. Le langage de programmation utilisé a aussi été développé spécialement (ROBOL/0).

Afin de communiquer ses états internes, le robot dans sa version de base est doté d'un synthétiseur vocal.

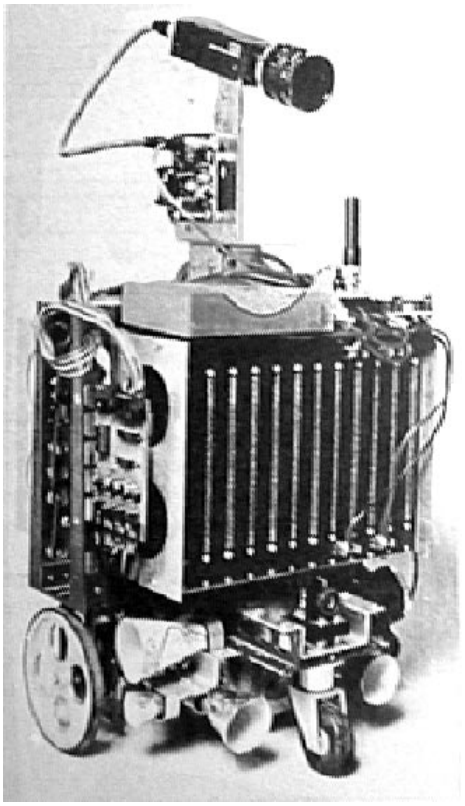


Figure 4: Le robot Yamabico

Ce robot reste donc d'assez grande taille et basé sur des outils spécifiques. Si ceci n'est pas gênant en ce qui consiste le système opératif, qui doit nécessairement supporter le temps-réel, ceci est plus gênant en ce qui concerne le langage de programmation, le ROBOL, qui limite la diffusion des résultats.

Enfin ce robot est essentiellement utilisé en mode auto-suffisant, à la fois énergétiquement et en puissance de calcul. Les outils d'analyse de l'état du robots sont, du moins pour la version de base, très limités.

2.2.1.2 Les robots du Massachusetts Institute of Technology

Le professeur Rodney Brooks est très probablement une des personnes les plus connues dans le domaine de la robotique mobile autonome. Dans son laboratoire, l'Artificial Intelligence Laboratory du Massachusetts Institute of Technology (MIT) il a créé, avec ses collaborateurs, un grand nombre de robots mobiles afin de prouver les principes de la *subsumption architecture*, décrite dans la section 1.6.2. Ces robots ont des formes, des fonctionnalités et des structures matérielles très diverses, pour pouvoir réaliser des expériences dans un large domaine d'applications, allant de l'exploration interplanétaire à la recherche de boîtes de Coca-Cola vides [Brooks90].

Le robot Allen est le premier de la série. Construit dans les années 86-87, il dispose d'odometrie et de capteurs à ultra-sons. Le comportement implémenté sur ce robot est un évitement d'obstacles associé à une exploration aléatoire.

Tom et Jerry sont deux robot conçus une année après Allen. Dotés de trois capteurs de proximité infra-rouge à sortie binaire, ces robots-jouets étaient contrôlés par une logique programmable à 256 portes. Le but était de prouver combien les structures de contrôle réalisées à l'aide de la subsumption peuvent être compactes.

Herbert est un robot bien plus complexe réalisé en 1988. Il dispose, du point de vue puissance de calcul embarquée, de 24 processeur 8 bits interconnectés par une connexion sérielle très lente (240 bits par seconde). Du point de vue moteur, Herbert, en plus du système de locomotion à roues, dispose d'un bras et d'un préhenseur qui lui permettent de saisir des objets. Du point de vue sensoriel, Herbert dispose de 30 capteurs de proximité infrarouge, de quelques senseurs simples sur le préhenseur, et d'un système de projection de plan laser pour la saisie de mesures de profondeur sur un angle de 60 degrés devant le robot. Ce dernier système permet de reconstruire une image de profondeurs de 256 x 32 points. Ce robot a été programmé pour ramasser des boîtes de Coca-Cola vides, ce qui nécessite une grande quantité de comportements allant de l'évitement d'obstacles à la reconnaissance des boîtes de Coca.

Gengis, développé en 1989, est le premier pas dans la direction des robots à pattes. Il dispose en effet de six pattes ayant deux degrés de liberté chacune. Le système de contrôle est complètement distribué sur 6 processeurs associés aux 6 pattes. Les capteurs dont il dispose sont des capteurs de force (12), six capteurs pyro-électriques (pour la détection de sources de chaleur), un inclinomètre et deux moustaches. Sa programmation a été poussée dans la direction de la coordination pour la marche, afin de réaliser des comportements comme le suivi de personnes. Ce robot existe aussi dans une version commerciale.

En 1989 a été développé Squirt, un robot miniature d'un volume d'un peu plus qu'un pouce cube. Ce robot a été pour le MIT un exercice de miniaturisation. En effet le robot est autosuffisant et dispose d'un microprocesseur à 8 bits, un capteur de lumière et deux microphones. Il a été programmé pour se cacher de la lumière et suivre un son particulier. Ce robot a été en partie la source d'inspiration pour le robot Khepera, développé en parallèle au LAMI et utilisé dans ce travail.

Tous ces robots sont programmés avec un langage et un environnement de pro-

grammation développés dans le même laboratoire. Le langage de programmation est le *Behavior language*, spécialement développé et adapté pour réaliser des programmes de contrôle basées sur la *subsumption architecture* [Brooks86]. Le compilateur qui supporte ce language, réalisé en lisp, est en effet un cross-compilateur qui génère du code pour le processeur du robot (souvent un ou plusieurs Motorola 68HC11, plus récemment des Motorola 68332). L'environnement de programmation permet essentiellement de télécharger le code et de monitorer des messages provenant du robot via la ligne sérielle, afin de visualiser une variable interne au robot. Dans la plupart des cas, toutefois, le robot est déconnecté de l'ordinateur hôte et dispose d'un affichage de quelques caractères ainsi que de quelques LEDs pour visualiser ses états internes.

2.2.1.3 Le robot Lola

Lola a été un des premiers robots développés dans le groupe du Prof. Steels, à Bruxelles. Ce robot et les expériences associées avaient été présentées dans le cadre de l'école NATO qui s'est déroulée à Trento [Steels95b]. De taille importante (environ 60 cm de diamètre pour un mètre de haut) ce robot se déplace avec une plateforme synchro-drive: Trois roues disposées sur un triangle équilatéral, toutes motrices, pouvant être dirigées ensemble dans n'importe quelle direction de façon synchronisée. Du point de vue des capteurs, Lola est équipé de détecteurs de collision, de capteurs de distance et de capteurs de son, qui lui permettent de détecter des sources sonores.

L'environnement du robot, essayant de recréer un écosystème artificiel dans lequel le robot doit évoluer, est doté de sources sonores représentant des sources de nourriture. Pendant les expériences, ce robot peut être suivi par un système de détection de la position absolue du robot dans son environnement. Ce système est composé d'une caméra avec une optique à grand angle fixée au plafond. La détection du robot par la caméra est facilitée par une puissante source lumineuse placée au dessus du robot. Un système similaire de mesure de la position du robot est décrit dans [Fleury93], qui utilise un motif particulier plutôt qu'une source lumineuse. Avec un temps de calcul supérieur aux 200ms, ce système peut détecter un robot sur une surface de 5 x 10 m avec une précision d'environ 5mm.

Dans le groupe du professeur Steels, le robot Lola a actuellement été remplacé par les robots LEGO, décrits plus loin.

Parallèlement à Lola, le groupe de Bruxelles a développé un language de programmation, appelé PDL [Steels92], qui permet une programmation facilitée de structures de contrôle temps-réel similaires à celles présentes dans la subsumption.

2.2.1.4 Autres systèmes

Beaucoup d'autres robots ont étés développés dans différents centres de recherche, et ce n'est pas le but ici d'en faire une liste exhaustive. Ces robots se différencient essentiellement par leur système moteur, leur capteurs et leur puissance de calcul. Ceux qui ont étés mentionnés ci-dessus couvrent une large palette des possibles combinaison. Pour la compléter convenablement il faudrait toutefois mentionner le robot KAMRO, développé à Karlsruhe [Lüth94]. Ce robot mobile de grosse taille pèse quelques centaines de

kilos et transporte deux bras manipulateurs de type PUMA 260, ayant chacun 6 degrés de liberté. Les préhenseurs de ce robot sont équipés d'un capteur de force/couple à six degrés de liberté. Au niveau des capteurs, trois caméras permettent la reconnaissance des objets manipulés et 16 capteurs à ultra-sons permettent une détection des obstacles. Il a été développé essentiellement pour des tâches d'assemblage automatisé nécessitant deux bras et une plateforme mobile. Celle-ci a la particularité d'être omnidirectionnelle: Quatre roues indépendantes et recouvertes, au lieu des pneus, de rouleaux placés à 45 degrés, permettent des déplacement et rotations dans n'importe quelle direction. Ce robot, comme beaucoup d'autres dans le domaine de l'automation, est associé à un logiciel qui permet de réaliser la simulation des actions du robot.

2.2.2 Les environnements disponibles commercialement

On présente ici quelques produits qui sont disponibles dans le commerce en tant qu'outil de recherche ou d'enseignement.

2.2.2.1 Le KitBorg

Le KitBorg est un robot mobile cylindrique de petite taille (15-20 cm) produit par Aleph Technologies et vendu essentiellement en France. Destiné principalement à l'enseignement de l'électronique, l'électrotechnique et la robotique, ce robot a aussi été utilisé dans des équipes de recherche [Bessière92]. Il est équipé de base d'un microcontrôleur NEC78310 (8/16 bits) et de 32k de RAM. Le déplacement du robot est effectué par deux roues situées sur les côtés du robot et contrôlées par deux moteurs à courant continu. Il dispose de 4 capteurs d'intensité lumineuse, un microphone pouvant détecter des claquements de main, deux capteurs à infrarouge, deux capteurs à ultra-sons pour la détection d'obstacles, ainsi que de deux détecteurs de collision. La programmation se fait ici aussi par téléchargement de code programmé en C ou en graphcet. Un dévermineur est disponible et il est possible de visualiser des informations au niveau d'un PC relié au robot par une ligne sérielle.

Figure 5: Le Kitborg. Son hauteur est d'environ 30 cm

2.2.2.2 Les robots de TAG

TAG est une entreprise anglaise qui vise plus particulièrement le monde de la recherche et de l'enseignement en robotique mobile et réseaux de neurones. Leurs produits sont des plateformes mobiles (robots Frank et Igor) ainsi que des cartes de contrôle neuronal analogique (N-Euro).

Frank est un robot à chenilles très similaire, dans sa structure et ses fonctionnalité, au robot Yamabico. Plus bas et plus long que ce dernier (50x24x24 cm), il est aussi composé essentiellement d'un fond de panier qui sert à loger les cartes de contrôle. Il dispose aussi de capteurs à ultra-sons, à infrarouge et tactiles. Son utilisation est donc similaire à celle du Yamabico. Aucun outil spécifique d'analyse ou de développement ne semble être disponible.

Igor (figure 6) est un robot à quatre pattes contrôlées pneumatiquement par 8 pistons. Chaque patte peut être levée et bougée selon l'axe d'avancement du robot. La taille de ce robot est réduite (18x20x21 cm) et il pèse 1.5 Kg. Il dépend d'une source d'air comprimée, mais les électrovannes de commande sont embarquées sur le robot. Une option permet d'embarquer une source d'air comprimé. Igor dispose de capteurs de contact au sol et de capteurs d'angles des pattes par rapport au corps du robot. Le tout peut être contrôlé depuis un PC.

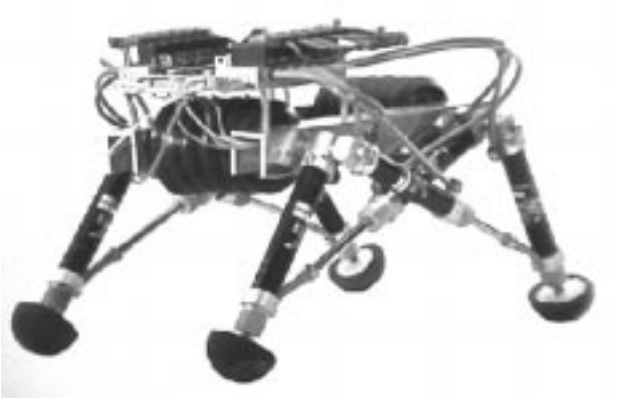


Figure 6: Le robot Igor

Ce robot est intéressant du fait qu'il permet d'expérimenter la marche dynamique et que son contrôle depuis un PC permet d'utiliser des outils standard pour la visualisation, la programmation et l'analyse. Il a le désavantage de dépendre d'une source d'air comprimé, mais ceci permet de garantir des performance intéressantes pour la taille réduite du robot.

2.2.2.3 Les robots de IS Robotics

IS Robotics commercialise une vaste palette de robots conçus au MIT, à roues et à pattes (voir aussi la section 2.2.1.2 *Les robots du Massachusetts Institute of Technology*). IS Robotics reprend les développements faits au MIT et les complète afin d'obtenir un produit. Malgré le nom d'une école prestigieuse comme le MIT accompagné du nom de R. Brooks, ces robots ne sont pas connus pour leur robustesse.

Parmi les robots à roues, le plus connu et le plus utilisé est probablement le R2 (figure 7). Ce robot, d'une hauteur d'environ 30-40 cm, se déplace en utilisant deux roues placées devant le centre de gravité et s'appuie sur une roue folle placée à l'arrière. Sur l'avant il est équipé d'une pince montée sur un axe vertical, ce qui lui permet de saisir des objets et les soulever. Il dispose de capteurs de proximité infrarouges et de capteurs de collision. La pince peut détecter la présence d'un objet par une barrière optique.

Du point de vue processeur, ce robot existe en deux versions, équipé d'un Motorola 68HC11 ou d'un Motorola 68332 (version R2E). L'environnement de programmation est basé sur le *behavior language* de Brooks. Il est disponible sur Macintosh et permet de visualiser une variable à la fois, transmise du robot à l'ordinateur hôte par une ligne sérielle. Le seul autre moyen de visualisation de l'état interne du robot est l'affichage LCD de quelques lignes dont le robot est doté. D'autres environnements de programmation (C par exemple) ne sont pas disponibles à ma connaissance.

Figure 7: Le robot R2 commercialisé par ISRobotics

2.2.2.4 Le Nomad

Nomadic est probablement une des entreprises les plus actives dans le domaine de la robotique mobile pour la recherche. Le robot le plus connu de cette entreprise est le Nomad 200.

Le Nomad 200, ayant un diamètre d'environ 60 cm, dispose d'un système moteur basé sur le principe synchro-drive. Au niveau sensoriel ce robot possède des capteurs de collision, des capteurs de proximité infrarouge et des capteurs de distance à ultra-sons.

Du point de vue processeur, le Nomad 200 dispose d'un PC complet. Cet ordinateur est souvent mis en communication avec un ordinateur hôte par une ligne sérielle radio.

Du point de vue de l'environnement de développement, Nomadic propose un simulateur du robot qui permet de préparer des expériences sans devoir mettre en route le robot. Ce simulateur est disponible sur SUN. Le robot peut ensuite être commandé par la ligne sérielle en utilisant un protocole de contrôle. Ceci permet de réaliser un relative-

ment bon environnement de travail, probablement le meilleur de ceux décrits dans cette section.

2.2.2.5 Les robots de RWI

RWI (Real World Interface) est une petite entreprise dont le patron est l'ancien initiateur des produits de Nomadic. Cette entreprise, qui met en premier plan la qualité et arrive a garder des prix compétitifs, est en forte croissance. RWI commercialise des robots très semblables au Nomad 200, basés sur des plateformes synchro-drive, ainsi qu'un robot très simple et de taille moyenne, le Pioneer, développé au SRI International [Nilsson84]. Ce dernier robot, qui pèse quelques kilos, est doté de deux roues latérales dont l'axe est décentré vers l'avant, ainsi qu'une roue folle à l'arrière. Les capteurs à disposition sont soit des capteurs à infrarouge, soit des capteurs à ultra-sons, au choix. Contrôlé par un microprocesseur 68HC11 qui s'occupe uniquement des capteurs et des roues ainsi que d'un protocole de commande, il est généralement équipé d'un PC portable placé sur le dos du robot, spécialement conçu à cet effet.

Ce robot est associé à un logiciel de navigation, appelé Saphira, qui permet la navigation dans des environnements dont on dispose d'un plan préalablement introduit dans le logiciel. Saphira est exécuté sur une station de travail et commande le robot via une connection radio.

2.2.2.6 Les robots LEGO

L'entreprise de jouets LEGO ne commercialise pas directement des robots. Elle propose uniquement une large palette de boites de construction technique, basées sur la très connue brique de base en plastique.

Plusieurs chercheurs utilisent les possibilités offertes par ce jeu de construction pour réaliser la structure physique de leur robot [Steels94] [Dennett91] [Jones93] [Smithers94] (figure 8). L'utilisation des LEGO permet en effet à n'importe qui de réaliser rapidement une structure physique, sans besoin de connaissances mécaniques particulières et avec un coût limité.

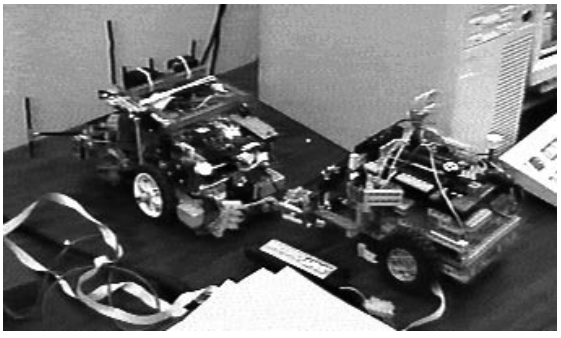
a.  b. 

Figure 8: Robots lego: exemple de forme générale (a.) et détail de construction (b.)

Pour contrôler cette structure physique, les chercheurs utilisent des cartes processeurs disponibles sur le marché (carte MiniBoard du MIT, utilisée par exemple dans

[Nolfi94]), ou développées expressément pour cette application [Smithers94]. Les capteurs utilisés pour ce type de robot sont les capteurs standard que l'on retrouve sur d'autres robots: capteurs de collision basés sur des microrupteurs, capteurs de proximité et de luminosité infrarouge et barrières optiques pour ne citer que les plus communs. Au niveau actionneurs, par contre, on préfère utiliser les actionneurs propres de LEGO à cause de leur facilité d'intégration dans la structure physique.

Cette approche à la robotique mobile donne à l'expérimentateur une très grande liberté d'action en ce qui consiste la morphologie du robot, mais pose aussi de gros problèmes:

- La structure de construction LEGO est basée sur un assemblage par emboîtement et est adaptée à une utilisation statique. Elle peut être utilisée pour des constructions ayant des parties mobiles, mais ceci doit être limité a un certain type de construction, dans laquelle l'effort est supporté par la structure et non par le frottement entre les éléments. Or ceci est une très grande limitation, qui rend les constructions LEGO très sensibles aux chocs et aux vibrations, car ce type d'effort n'a pas de direction préférentielle.
- Les actionneurs LEGO sont conçus pour des jouets, utilisés essentiellement à vitesse constante, sans souci pour la vitesse même du moteur, pour le couple ou pour la position de l'effecteur. Ceci limite considérablement les possibilités de contrôle du robot, forçant les chercheurs à utiliser des vitesses fixes, non réglées, sans aucune possibilité de régler la position de l'effecteur sinon que par des capteurs externes, normalement assez difficiles à intégrer dans la structure LEGO et nécessitant des connaissances plus spécifiques.

Les environnements de programmation sont liés au type de carte processeur embarquée. Généralement il s'agit de cross-compilateurs C sur PC. Afin de suivre et analyser le comportement du robot il y a principalement deux approches: La première (utilisée par exemple par le groupe de Steels) consiste à équiper le robot d'une matrice de LEDs qui servent à visualiser l'état interne. La deuxième approche (utilisée par Smithers) consiste à utiliser des PC très compacts qui peuvent être embarqués sur le robot. Ces deux approches limitent fortement l'observation des paramètres du système de contrôle du robot lors de l'expérimentation.

Enfin l'équipe de L. Steels [Steels94] se sert d'une caméra placée au dessus de l'environnement d'expérimentation pour mesurer le déplacement du robot. L'image vidéo est enregistrée, puis élaborée pour en extraire la trajectoire du robot. L'élaboration se fait une fois l'expérience terminée.

2.2.2.7 Fisher Technik

Fisher Technik est un autre jouet de construction bien connu et est utilisé pour les mêmes raisons et les mêmes buts que le LEGO. Un robot construit en utilisant cette technique est présenté dans la figure 9.

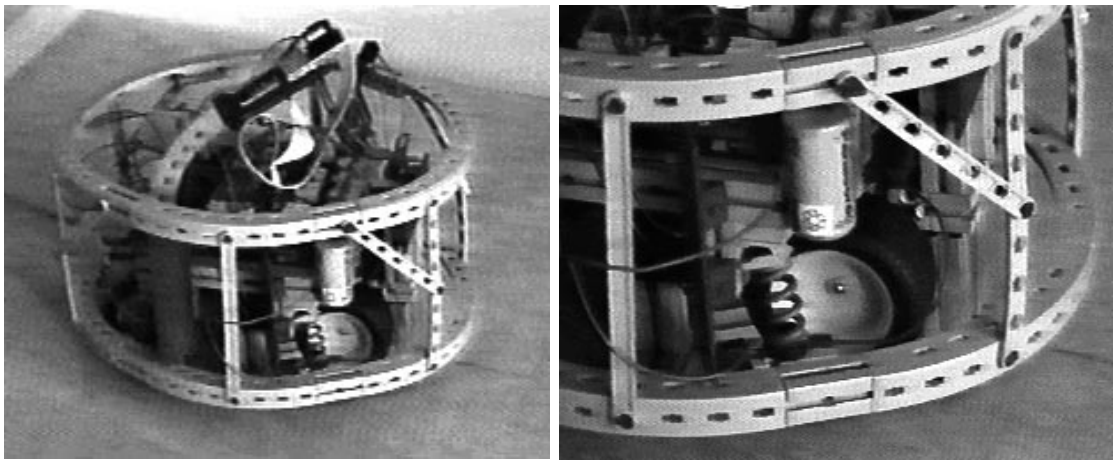
Du point de vue de la construction, Fisher Technik est beaucoup plus robuste que

LEGO: les pièces, toujours en plastique, s'imbriquent par des fixations en queue d'hirondelle. Les axes de transmission sont métalliques et les engrenages se fixent sur les axes par des mors coniques. Ceci permet de réaliser des constructions très stables et qui supportent des efforts beaucoup mieux que celles réalisées avec du LEGO.

De plus Fischer Technik propose des boîtes de construction comportant des capteurs de différents types (de luminosité, de champ magnétique), qui peuvent être utilisés dans des robots ayant des fonctionnalités très simples.

Si le désavantage de manque de robustesse de LEGO disparaît presque complètement dans Fischer Technik, le désavantage concernant les actionneurs reste identique. Comme LEGO, Fischer Technik propose des actionneurs qui s'intègrent parfaitement dans la construction mais qui se révèlent pratiquement inutilisables pour la robotique.

Au niveau des environnement de développement, déverminage et analyse, la situation est identique à celle des robots LEGO.



a. b.

Figure 9: Exemple de robot construit avec du Fischer Technik (a.) et détail de construction (b.)

2.2.2.8 Les outils logiciel

En plus des outils liées à un robot particulier, il existe un grand nombre d'autres logiciels commerciaux qui peuvent être utilisés pour la visualisation en temps réel ou l'analyse de résultats obtenus avec les robots mobiles.

Parmi les logiciels spécialisés dans la visualisation de données provenant de mesures externes à l'ordinateur (appelés aussi logiciels d'instrumentation) un des plus connus est LabVIEW®. Ce logiciel est un produit de National Instruments. Le but de ce logiciel est de permettre de programmer de façon simple et efficace des interfaces entre un utilisateur et une installation à surveiller. National Instruments met donc à disposition, d'une part, une série d'outils pour l'accès à de l'équipement extérieur, comme des interfaces GPIB (IEEE488), RS232, des cartes d'acquisition numérique ou analogique, des cartes de contrôle numérique ou analogique, etc. D'autre part LabVIEW® dispose d'une large librairie d'outils de traitement, visualisation et interaction avec les données saisies et

envoyées sur l'installation externe.

Dans LabVIEW®, les outils de visualisation et contrôle s'appellent *Virtual Instruments* (VI). Un VI est composé d'un panneau frontal, qui joue le rôle d'interface avec l'utilisateur, et d'un diagramme, qui réalise les fonctionnalités de l'instrument. Sur le panneau peuvent être disposé des instruments de visualisation comme des interfaces de contrôle graphiques, comme le montre la figure 10. Chaque objet graphique est obtenu par un choix dans le menu des objets de contrôle. Les attributs de chaque objet peuvent être modifiés en agissant directement sur l'objet ou en choisissant des paramètres dans un menu déroulant. Chaque objet présent sur le panneau porte un nom et correspond à un terminal de données au niveau du diagramme (voir figure 11). Ces terminaux peuvent être câblés entre eux en utilisant une librairie de fonctions disponibles sous forme d'icônes.

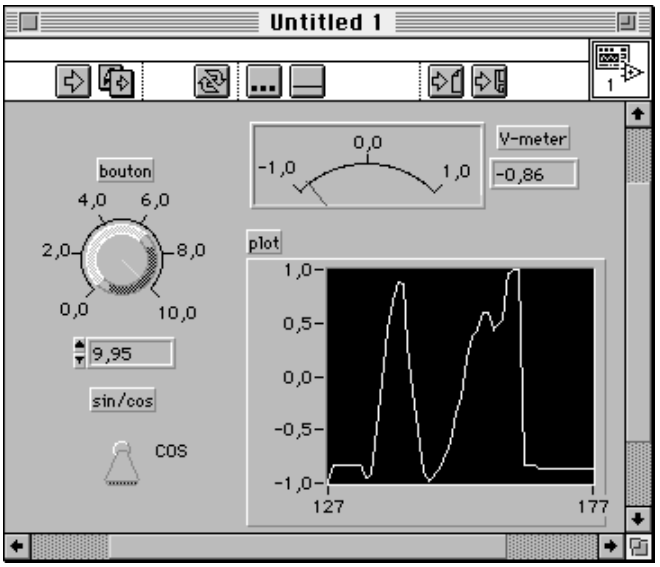


Figure 10: Exemple d'interface utilisateur crée avec LabVIEW®

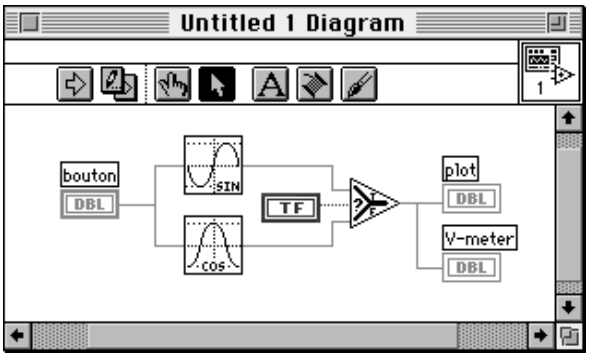


Figure 11: Diagramme de fonctionnement d'un instrument virtuel sous LabVIEW®: La valeur numérique obtenue du bouton du panneau frontal est traitée par des fonctions sinus et cosinus. Le résultat d'un de ces deux opérateurs, selon la valeur booléenne donnée par le commutateur *sin/cos*, est affiché sur les deux affichages *V-meter* et *plot*

La mise en place des objets sur le panneau frontal, leur modification ainsi que toutes les opérations au niveau de la description du fonctionnement de l'instrument dans le diagramme sont réalisés de façon graphique et interactive. Sans aucune connaissance préalable de langages de programmation il est donc possible de réaliser des acquisitions de données, leur traitement ainsi que la visualisation et le contrôle éventuel de paramètres de l'installation en temps réel. Lors de calculs complexes, il est possible d'inclure dans LabVIEW® des modules écrits en C et visualisés dans LabVIEW® sous forme d'icône.

En robotique mobile il est possible d'utiliser ce potentiel pour la programmation, la supervision et la visualisation des résultats d'algorithmes de contrôle. En ayant un robot qui peut être contrôlé par un protocole implémenté sur une ligne sérielle RS232, toutes les fonctionnalités du robot peuvent être très intuitivement mises à disposition de l'utilisateur sous forme d'icônes. Les possibilités graphiques de LabVIEW® peuvent donc être exploitées pour visualiser l'état des capteurs du robot, une interface peut permettre de contrôler la vitesse d'avancement du robot, etc. La grande limitation de ce logiciel est sa lenteur lors de calculs ou affichages importants.

D'autres outils d'analyse, comme Mathematica ou MatLab, permettent une bonne analyse des données récoltées pendant la simulation, mais ne permettent pas un contrôle direct du robot, ni une analyse des données en temps réel. Wolfram Research, producteur de Mathematica, et toutefois en train de développer une interface entre Mathematica et LabVIEW®, ce qui permettrait l'accès aux fonctionnalités du robot directement depuis Mathematica.

2.3 L'outil Khepera

Au Laboratoire de Microinformatique (LAMI) de l'Ecole Polytechnique Fédérale de Lausanne (EPFL), il y a toujours eu un intérêt pour les robots mobiles qui associent très étroitement la mécanique (bien maîtrisée par André Guignard), l'électronique et les capteurs (spécialité de Edo Franzi), les microprocesseurs et la programmation temps réel (domaine de base du laboratoire). En 1990, sur la lancée du MIT qui présentait Squirt, un robot d'un pouce cube (voir aussi la section 2.2.1.2), le LAMI a démarré un projet ayant pour but de réaliser un robot mobile plus petit que celui du MIT. Il en a fait un travail de semestre et un de diplôme, réalisés par Kaspar Suter et terminés en 1991. Dans cette même période, le LAMI a participé à un projet de recherche sur l'intelligence artificielle en robotique (PNR23) dans lequel il avait pour tâche d'analyser les applications des réseaux de neurones artificiels (RNA) dans ce domaine. Après une analyse des applications du domaine industriel (bras manipulateurs) l'intérêt s'est porté sur la robotique mobile, un domaine de recherche dans lequel les réseaux de neurones avaient encore des chances de trouver une place. De ces deux activités, le robot miniature et la recherche sur les réseaux de neurones en robotique, est née l'idée d'utiliser un robot miniature pour faire les expériences prévues. Finalement, Khepera a été conçu comme un outil de recherche de base avec des possibilités limitées mais suffisantes pour mettre en évidence les potentialités des algorithmes de contrôle mis en oeuvre sur un robot mobile réel.

2.3.1 La miniaturisation

La réduction de taille d'un robot amène beaucoup d'avantages et peut-être une innovation dans le domaine des robots mobiles utilisés dans la recherche pour des raisons variées:

1 La taille réduite du robot permet de le manipuler plus facilement. Les déplacements se font facilement et l'arrêt du robot en cas d'urgence se fait à la main, si nécessaire.

2 Les lois de similitude donnent à un robot de petite taille une résistance mécanique relative plus élevée. Si l'on réduit, de façon homothétique, un élément porteur dimensionné pour porter une charge donnée, la réduction d'un facteur 2 de la charge nécessite un élément porteur qui peut être réduit d'un facteur 2.83. L'élément porteur, si réduit d'un facteur 2, peut donc porter 1.26 fois la masse réduite du même facteur. Ceci est dû au fait que si la masse est réduite proportionnellement au cube des longueurs, la résistance des éléments porteurs de cette masse se réduit proportionnellement au carré des longueurs. Ce fait rend un robot réduit homothétiquement plus résistant aux chocs, par exemple.

3 Cette même résistance mécanique permet de réaliser le robot avec des techniques plus simples et moins coûteuses, en utilisant des connecteurs électroniques comme support mécanique, par exemple.

4 La taille des environnements est réduite, ce qui signifie un gain de place, et la réalisation d'environnements de test se trouve nettement facilitée. Pour des robots de taille standard (50 cm de diamètre), il est souvent nécessaire d'utiliser un grand local, et sa préparation nécessite à la fois du temps et de l'argent.

5 La hauteur disponible au-dessus de l'environnement est proportionnellement très grande. Ceci permet, par exemple, de suspendre un câble connecté au robot qui ne le gêne pas dans ses mouvements. D'autres systèmes de référence (une caméra, par exemple) peuvent être placés à une distance verticale relative considérable.

6 Le transport, les démonstrations et les échanges se trouvent facilités par la petite taille, la résistance et le prix bas.

Naturellement il y a également des désavantages à la miniaturisation:

1 Les lois de similitude montrent que la réduction homothétique de taille n'est pas favorable aux actionneurs électriques qui sont basés sur l'électro-magnétisme. Le rendement d'un moteur électrique, réduit homothétiquement d'un facteur 20 (de 100 mm à 5 mm, par exemple), peut chuter de 90% à 2%. De plus, les problèmes de stockage de l'énergie (batteries) deviennent problématiques.

2 La réalisation électronique et surtout mécanique se trouvent limitées. Ce fait restreint le choix des capteurs et actionneurs, limitant ainsi les possibilités d'expérimentation.

3 Il n'est pas possible de faire face aux objets que le robot rencontrera dans sa vraie implémentation pour l'application. On sera confronté à des objets de taille et complexité réduite, suivant les contraintes physiques imposées par les lois de similitude.

Malgré ces inconvénients, la miniaturisation reste très intéressante. Le premier et le deuxième désavantages, de type technique, ne sont que peu contraignants pour des expériences de principe qui visent des tâches relativement simples, comme l'évitement d'obstacles et la navigation. Il est à noter que les expériences visées dans ce travail sont de ce niveau. Ainsi nous nous trouvons aussi moins touchés par le troisième désavantage, tout en restant dans un environnement réel.

En résumé, la miniaturisation semble apporter un vrai avantage à l'expérimentation en robotique réelle. La section suivante illustre la mise en oeuvre de ce principe dans le robot mobile miniature Khepera.

2.3.2 Le robot mobile Khepera: sa structure et son fonctionnement

Le niveau de miniaturisation choisi a conduit au développement de Khepera, un robot cylindrique de 55 mm de diamètre et de hauteur variable selon la configuration. Ce robot a été développé par Edo Franzi, André Guignard et moi-même.

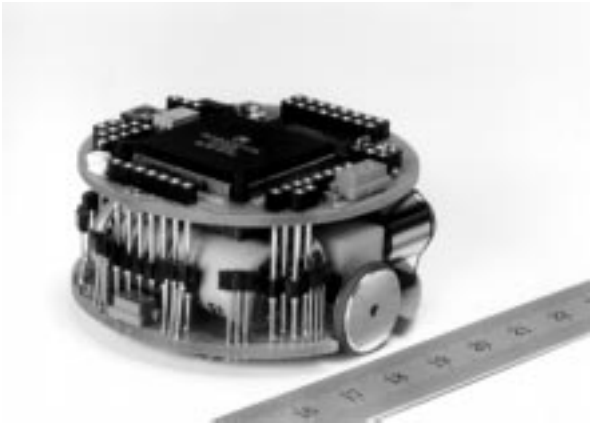


Figure 12: Le robot mobile miniature Khepera dans sa configuration de base

Selon le classement proposé par Paolo Dario [Dario92], la dénomination pour ce robot est *robot miniature*. L'ordre de grandeur de ses dimensions (3 à 5 cm) est le plus petit que l'on peut atteindre avec des composants standard du marché, tels que les moteurs et les microcontrôleurs, les batteries et les capteurs. En restant à ces dimensions plutôt que descendre encore plus bas, on arrive encore à embarquer un processeur performant (M68331, 32 bits cadencé à 16MHz). En outre, cette taille permet de réaliser un grand environnement sur une surface limitée. Pour Khepera, en effet, une table normale de 0.9 x 1.8 mètres représente la même surface de travail relative qu'un terrain de tennis pour un robot d'un diamètre de 55 cm, comme le Nomad 200, par exemple. La hauteur dont on dispose, avec Khepera, au dessus de la table dans un local normal, correspondrait à une maison de 4 étages pour le Nomad 200. Cette hauteur relative permet, par exemple,

de relier le robot à un dispositif fixe par un câble suspendu, sans déranger les déplacements du robot. Grâce à cette place on peut aussi positionner une caméra qui observe l'environnement sans avoir besoin d'objectifs à grand-angle qui causeraient des fortes déformations (comme dans [Steels94]).

Comme on peut le voir dans la figure 12, Khepera, dans sa configuration de base, est composé de deux couches correspondantes aux deux circuits imprimés qui le constituent: le circuit sensori-moteur et le circuit CPU, qui inclut le microprocesseur qui gère le robot. Les deux circuits sont reliés électriquement par un système de connecteurs qui assure aussi la liaison mécanique.

Le système moteur est basé sur deux roues latérales et deux points d'appui à l'avant et à l'arrière. Cette configuration permet de faire face à des obstacles géométriquement très complexes grâce au fait que le robot peut tourner sur place sans se déplacer. Ce type de locomotion conserve la structure que l'on trouve dans les animaux et qui est basé sur des actionneurs latéraux et symétriques. Ceci n'est pas le cas d'autres structures comme par exemple le synchro-drive, utilisé dans [Franceschini92] et basé sur trois roues toutes motrices et orientables, ou l'omni-drive, utilisé dans [Lüth94] et basé sur quatre roues motrices indépendantes composées de rouleaux placés à 45° par rapport à l'axe de la roue.

Le système sensoriel du robot de base, placé sur le circuit inférieur, est très simple: 8 capteurs de proximité à lumière infrarouge sont disposés sur la périphérie du robot selon la distribution donnée en figure 13. Cette disposition donne plus de sensibilité à un côté du robot, qui est considéré comme étant le devant.

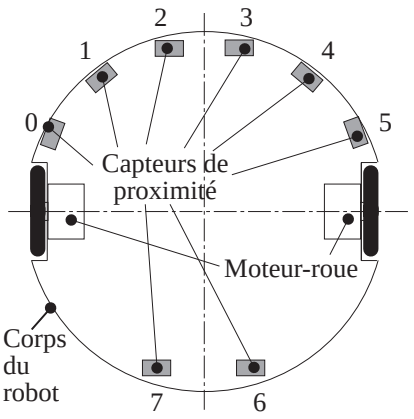


Figure 13: Disposition des capteurs de proximité et des deux roues

Les capteurs de proximité sont composés d'un émetteur et d'un récepteur de lumière infrarouge. Ils permettent de détecter la présence d'obstacles par la réflexion de lumière de ces derniers. Le fonctionnement est simple et se décompose en deux phases principales. En un premier temps, l'émetteur est laissé éteint et on mesure la luminosité ambiante grâce au récepteur (figure 14). Dans une deuxième phase, on allume l'émetteur et on mesure à nouveau la lumière sur le récepteur. La différence entre les deux mesures effectuées est due à la réflexion des objets entourant le robot.

La valeur finale obtenue de cette différence dépend essentiellement de l'intensité de la lumière réfléchie, qui, à son tour, est dépendante du carré du double de la distance de l'objet (figure 15). D'autres facteurs interviennent dans cette mesure, tels que la largeur des obstacles (influence sur la taille de la surface de réflexion), la luminosité ambiante (fait varier le gain du récepteur), la couleur et le type de surface des obstacles (effet sur le facteur de réflexion). Il ne s'agit donc pas d'un capteur de distance, mais bien d'un capteur de proximité ayant des réponses variables selon les obstacles rencontrés.

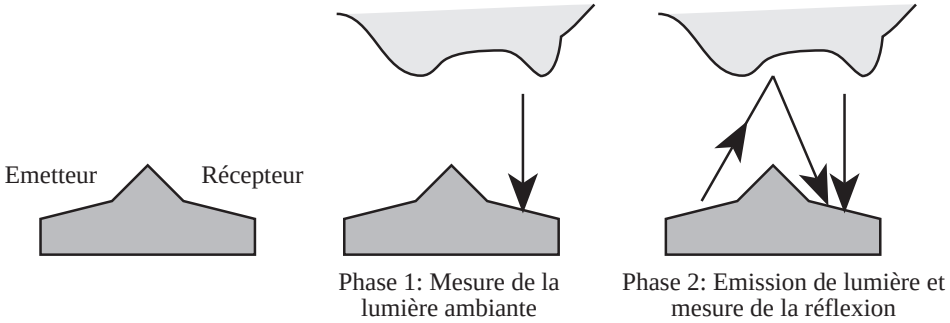


Figure 14: Structure et fonctionnement d'un capteur de proximité à réflexion

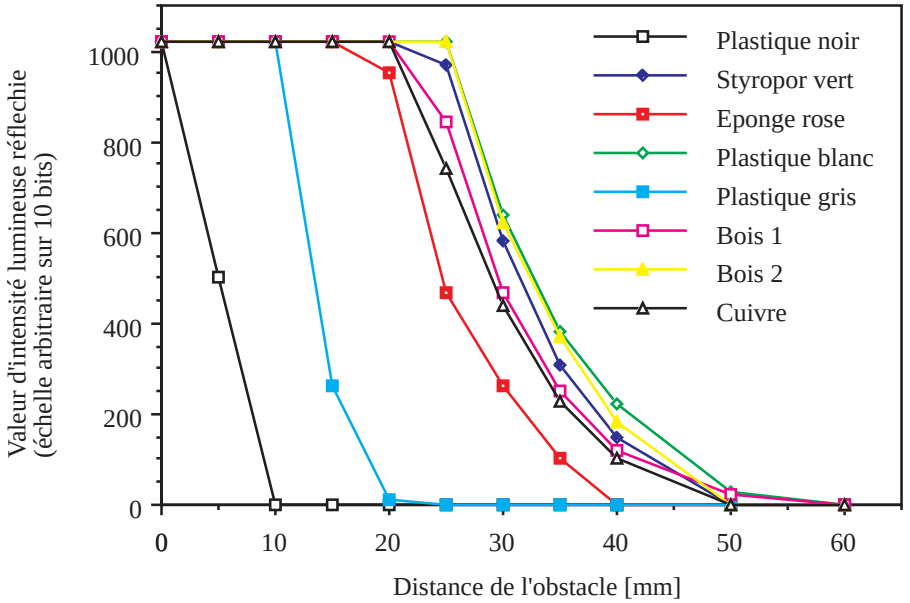


Figure 15: Caractéristiques de réponse des capteurs de proximité en fonction du matériel constitutif de l'obstacle

Les fonctionnalités de base de ce capteur permettent de l'utiliser aussi comme simple capteur passif de lumière infrarouge. Dans Khepera, l'information sur la lumière ambiante obtenue lors de la première phase de mesure de proximité n'est pas uniquement utilisée pour la soustraction décrite plus haut mais est aussi mise à disposition de l'utilisateur. Il est ainsi possible de détecter des gradients de lumière, la direction des sources lumineuses, etc.

En outre, sur le circuit sensori-moteur sont placés les accumulateurs NiCd d'une capacité de 110mAh et qui permettent au robot d'être autosuffisant énergétiquement pendant une durée d'environ 30 à 45 minutes. Un connecteur permet de recharger ces accumulateurs avec un chargeur externe. A travers ce connecteur, le chargeur a également accès à une mesure de la température des accumulateurs.

Le circuit CPU abrite le processeur du robot, du type Motorola 68331. Ce type de microcontrôleur est doté de façon interne de périphériques tels que timers, ports série synchrones et asynchrones, etc., sans pour autant permettre une utilisation en *single-chip*. Il nécessite, en effet, de la mémoire externe, dans notre cas une EEPROM de 128kBytes et une RAM statique de 256kBytes. Un convertisseur A/D, également sur ce circuit, permet l'acquisition des valeurs analogiques provenant des capteurs du circuit sensori-moteur. Un connecteur permet enfin de disposer d'une ligne série de type RS232 mais aux niveaux TTL. Sur le même connecteur, on dispose d'une ligne supplémentaire qui permet d'alimenter le robot depuis une source extérieure.

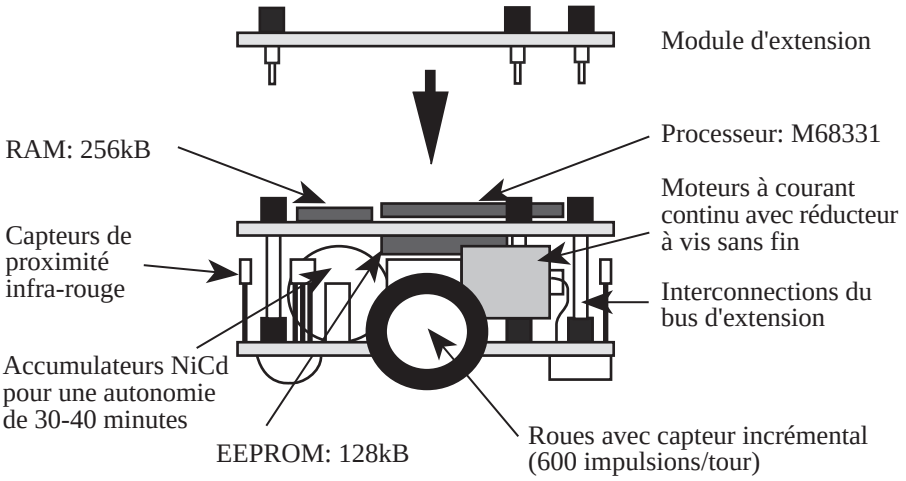


Figure 16: Structure du robot Khepera

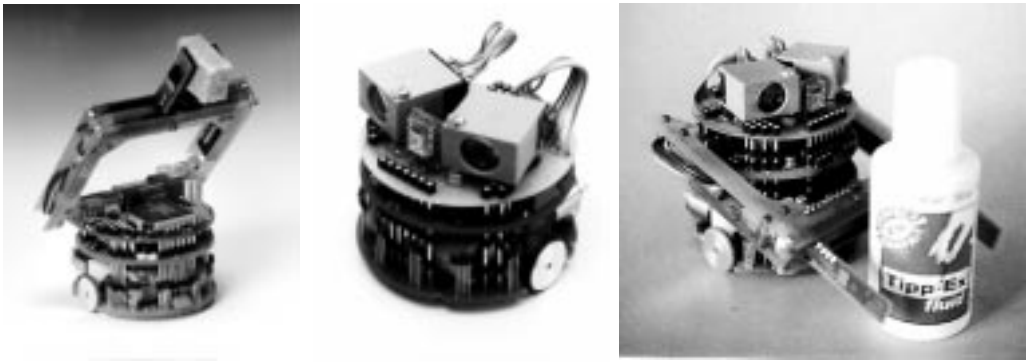


Figure 17: Possibilités d'extension, de gauche à droite: Khepera avec une extension *pince*, avec une extension *vision linéaire* et avec les deux types d'extensions simultanément

La liaison électrique entre le circuit sensori-moteur et le circuit CPU se fait par une série de connecteurs, qui font en même temps liaison mécanique. Ces connecteurs traversent les circuits et permettent un enfichage sur plusieurs couches. Des extensions réalisées sont, par exemple, le module *pince* et le module vision linéaire, les deux illustrés dans la figure 17.

Les extensions utilisées pour ce projet ont été le module KPS (de *Khepera Positioning System*) pour la mesure de la position et de l'orientation du robot, le module *pince* ayant deux degrés de libertés et le module de vision linéaire.

Le module KPS permet la mesure de la position absolue du robot à partir d'un système de balayage laser placé au dessus de la zone de travail comme l'illustre la figure 18.

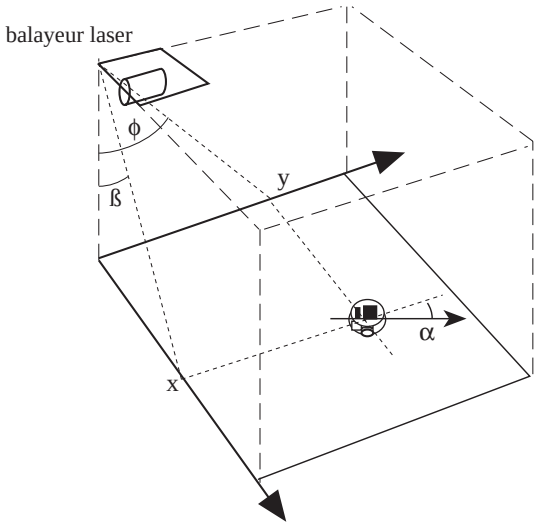


Figure 18: Système de balayage laser et indication des paramètres que le robot peut mesurer

Le système de balayage génère des plans laser qui balayent selon deux directions orthogonales le plan sur lequel se trouve le robot. Ce balayage s'effectue avec une fréquence d'environ 25-30 Hz. Lors du passage du plan laser sur la tourelle d'extension KPS située sur le robot Khepera, une photodiode permet de détecter le passage du laser et ainsi de calculer les angles ß et ϕ se trouvant entre le plan laser et le système de référence du système de balayage. En connaissant la position du système de balayage, il est facile de calculer la position absolue du robot. En plaçant deux photodiodes détectrices sur le robot, le calcul de leur position absolue et la connaissance de leur position par rapport au robot permet de calculer l'orientation α du robot. Enfin la séquence de balayage a été conçue pour contenir en elle-même toutes les informations de position. Grâce à cela, chaque tourelle de détection peut mesurer, de façon complètement autosuffisante et en temps réel, sa position absolue sur le plan de travail.

2.3.3 L'environnement de travail

Le fait d'avoir un robot, aussi performant et modulaire que l'on veut, n'est pas suffisant pour permettre de faire des expériences intéressantes. Les outils de conception

comme ceux d'analyse du travail effectué jouent un rôle très important. Les outils de conception sont assez répandus et relativement bien développés. Par exemple les supports logiciel aux différentes méthodes de conception [Brutzman95][Michel96] ainsi que les langages de programmation comme le *behavior language* [Brooks86], *PDL* [Steels92], *ALFA* [Gat91], *REX* [Kaelbling86] ou *BART* [Kahn91]. Les outils d'analyse, par contre, sont souvent limités, voir inexistants. Vu l'intérêt, lors de ce travail, d'une approche plus centrée sur l'autonomie et proche du monde biologique, il est apparu important de mieux développer les outils d'analyse, tout en utilisant des outils de conception performants.

Les outils de conception dont nous nous sommes servis dans ce travail sont les outils de conception logiciel habituels, tels que les environnements de compilation. La programmation des aspects de contrôle de haut niveau a toujours été faite en C, alors que les aspects de bas niveau et strictement temps-réel (régulateurs, échantillonnage des capteurs, etc.) ont été programmés en assembleur par Edo Franzi. Le choix du language C a essentiellement été fait à cause de l'usage répandu de ce language et des possibilités d'échanges scientifiques qu'il offre.

Parmi les outils de conception figure souvent le simulateur du robot. Cet outil permet de faire des études de faisabilité sans avoir besoin d'utiliser le robot et se justifie lorsque l'utilisation du robot physique est sensiblement plus compliquée, plus chère ou implique des durées d'expérimentation excessives. Ceci est effectivement le cas pour des robot de grosse taille comme ceux présentés dans la section 2.2.2 *Les environnements disponibles commercialement*. Dans le cas du Khepera, pourtant, la simulation perd beaucoup de son attrait, car le robot est maniable, bon marché et peut être relié à une interface très similaire à celle d'un simulateur. De plus, une partie des expériences réalisées avait déjà fait l'objet d'études en simulation. Pour ces raisons, pour celles déjà énumérées dans la section 2.1 *Les robots simulés et les robots réels* et afin de concentrer ce travail sur l'aspect de l'interaction robot (réel) - environnement (réel), nous n'avons pas développé de simulateurs.

Si les outils de conception sont essentiellement de nature logicielle, les outils d'analyse comportent aussi une partie matérielle. La conception est principalement un travail de programmation du robot lui-même, qui ne nécessite normalement que quelques petites modifications accessoires du matériel. L'analyse, par contre, impose que l'on fasse physiquement des mesures sur le robot, comme repérer sa trajectoire ou son passage par un certain endroit. Or, comme on l'a vu dans la section précédente (2.2), il existe très peu d'outils standard pour réaliser ces mesures. D'autres mesures peuvent être faites au niveau logiciel, par exemple en ce qui concerne les états internes du robot. Enfin les outils d'analyse doivent proposer une large palette d'outils mathématiques et de visualisation.

En ce qui concerne les outils matériels d'analyse, le plus grand effort a été fait sur le système KPS décrit dans la section 2.3.2 *Le robot mobile Khepera: sa structure et son fonctionnement*. Ce système de repérage a été conçu et utilisé uniquement comme outil d'analyse de la position du robot, ses données n'étant jamais mises à disposition du système de contrôle du robot. L'avantage de ce système est d'être léger en temps de calcul et

relativement bon marché par rapport à des systèmes comme celui utilisé par Luc Steels [Steels94] tout en garantissant un fonctionnement précis et temps-réel.

Au niveau logiciel il nous est apparu très important de pouvoir visualiser en temps-réel les mécanismes et les résultats analysés. Nous nous sommes donc concentrés sur des systèmes de visualisation temps réel comme LabVIEW® [LabVIEW96] et PackLIB [Cheneval95], tout en utilisant des outils mathématiques répandus comme Mathematica®, produit par Wolfram Research [Wolfram92] ou des tableurs connus comme Excel®, de Microsoft.

LabVIEW® est un logiciel commercial d'instrumentation développé et vendu par l'entreprise National Instruments. Il a été conçu principalement comme outil de création d'*instruments virtuels* (appelés VI, de *Virtual Instruments*). Un VI a comme but de piloter une installation, rassembler des données mesurées, les mettre en forme et les afficher dans un format compréhensible pour l'utilisateur. Les fonctionnalités de base de LabVIEW® permettent ainsi de créer des interfaces de commande, de commander des installations, de saisir des données, de faire des opérations sur ces données et de les afficher sous les formes les plus diverses.

L'intérêt d'utiliser LabVIEW® dans le cadre de la robotique mobile réside dans le fait que les opérations, indiquées ci-dessus, sont pratiquement les mêmes que celles utilisées par l'expérimentateur en robotique mobile. En effet, dans ce cas, il s'agit de contrôler une boucle entre les capteurs du robot (mesures) et les actionneurs du robot (pilotage de l'installation) en passant par un traitement des données. L'aspect d'interface avec l'utilisateur est fort utile, car il permet de suivre le détail de l'expérience de façon interactive.

Le désavantage principal de ce logiciel réside dans le fait que la partie du traitement des données n'est que peu développée. Or, si les fonctionnalités sont suffisantes pour de l'instrumentation, elles sont insuffisantes pour l'algorithmique plus poussée présente en robotique mobile. En plus, du coté des interfaces, on dispose de nombreux outils pour visualiser et agir sur des valeurs booléennes ou scalaires, mais très peu de possibilités sont offertes pour traiter des données sous forme vectorielles ou matricielles, par exemple. Finalement, la rapidité d'exécution est décevante. Dans le domaine de l'instrumentation, cet aspect peut être négligé à cause des faibles traitements nécessaires.

Un environnement qui s'adapte mieux aux nécessités de recherche évoquées ci-dessus est PackLIB. Cet environnement a été développé avec le but de permettre la visualisation de résultats obtenus par des algorithmes "gourmands" en temps de calcul, en particulier les réseaux de neurones. Dans cet environnement, on dispose en effet d'outils de visualisation très performants, tout en offrant la possibilité d'écrire la partie de calcul pur en C (voir [Cheneval95] pour plus de détails).

Pour pouvoir utiliser ces outils en temps réel et de la façon la plus efficace possible, nous avons adopté la configuration de travail décrite en figure 19 qui exploite la miniaturisation de Khepera. Dans cette configuration, le robot est en effet placé dans un environnement qui se trouve sur une table à côté d'une station de travail. Le robot est relié à la station de travail par un câble qui supporte une liaison sérielle pour la transmission des

données et l'alimentation du robot afin de ne pas dépendre des accumulateurs. La fonctionnalité visée par cet équipement est de disposer d'un robot fonctionnant en temps réel dans un environnement réel, tout en pouvant utiliser l'interface de la station de travail (écran, audio, souris, clavier) pour pouvoir suivre, analyser et interagir avec l'expérience de la façon la plus aisée. Si l'interface utilisateur est clairement située au niveau de la station de travail, et le système de contrôle temps-réel est nécessairement placé au niveau du robot, le système de contrôle comportemental peut être placé aux deux niveaux. Au niveau du comportement d'un robot mobile, les constantes de temps sont en effet assez longues, ce qui permet un contrôle via la ligne sérielle, vrai "goulet d'étranglement" du système.

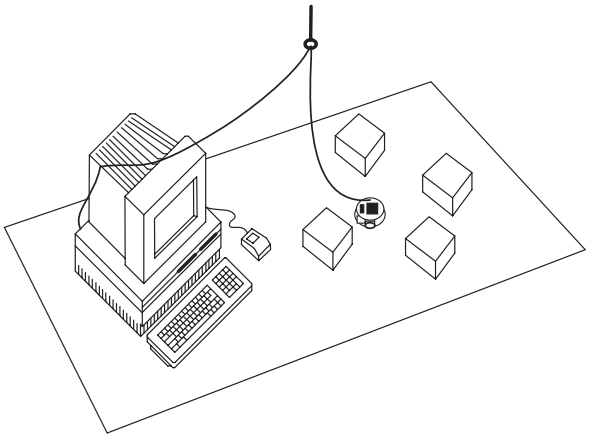


Figure 19: Environnement de travail comprenant un robot Khepera dans son environnement, relié à une station de travail utilisée pour le contrôle, le suivi et l'analyse de l'expérience

2.3.4 Les simulateurs de Khepera

Lors du développement de Khepera, comme déjà mentionné plus haut, aucun simulateur de ce robot n'a été prévu. Khepera a été développé dans l'idée de permettre une recherche plus approfondie et complémentaire à celle faite en simulation. Entre la première phase de développement et maintenant, un grand nombre d'universités et de centres de recherche ont acheté Khepera. C'est au niveau de ces utilisateurs, pour des raisons de manque de robots et surtout pour des études de base d'algorithmes à fonctionnement lent (essentiellement génétiques) que plusieurs groupes ont développés des simulateurs qui émulent les fonctionnalités de base de Khepera. Trois de ces simulateurs sont opérationnels:

- KhepSim de l'université de Rome, développé par Stefano Nolfi, permet de simuler Khepera équipé d'un module pince. Ce simulateur a été crée pour PC et ensuite porté sur SUN. Le fonctionnement du simulateur est basé sur des données mesurées sur un robot réel. A partir de ces données, le simulateur s'occupe de reconstruire les réponses sensorielles que le robot rencontrerait dans une situation donnée. Les structures de contrôle mises en oeuvre sont basées sur des réseaux de neurones. Le simulateur permet de visualiser, en plus de la position du robot dans un environnement, les différentes activités des neurones utilisés pour le contrôle. Stefano Nolfi utilise

ce simulateur pour effectuer, de façon accélérée, l'évaluation des capacités d'individus soumis à des algorithmes génétiques [Nolfi94][Nolfi95].

- KhepSim de l'université de Sussex, développé par Nick Jackobi, permet de simuler un Khepera dans sa configuration de base [Jakobi94] [Jakobi95]. Légèrement moins réaliste dans la gestion des mesures sensorielles et motrices que celui de Nolfi, ce simulateur réalisé sur SUN a une très bonne interface graphique qui permet de visualiser le déplacement du robot ainsi que les activités internes au système de contrôle. En plus du fait d'être modulaire et donc un peu plus flexible que le simulateur de Nolfi, il permet de générer automatiquement du code qui peut être téléchargé dans le processeur embarqué du Khepera. Il est utilisé par le groupe de Sussex pour des recherches du même type que celles faites par Nolfi.
- Sim de l'université de Nice, développé par Olivier Michel [Michel96], est similaire au simulateur de Sussex pour ce qui est de l'interface utilisateur et de l'ouverture à toute sorte d'expérimentation, tout en se basant sur des données provenant de mesures faites sur un robot réel comme pour le simulateur de Rome. De plus, il permet de simuler plusieurs robots. Au lieu d'être en mesure de créer du code téléchargeable, Sim permet d'utiliser au choix un robot simulé ou un Khepera connecté à une ligne sérielle. Ce dernier simulateur a en plus l'avantage d'être dans le domaine public, bien déverminé et muni de documentation. Des versions pour Macintosh et PC sont en cours de développement.

2.3.5 De Khepera aux applications

Comme il a été mentionné plus haut, Khepera est essentiellement un outil de base pour la recherche. Son utilisation permet de passer de la simulation au monde réel, en se penchant ainsi sur les problèmes d'interaction avec le monde physique. La taille de Khepera ne permet pas d'utiliser des capteurs ou des actionneurs très sophistiqués, encombrants ou consommant beaucoup d'énergie (mesure de distance au laser, capteurs ultrasons multiples, bras manipulateurs à 3-4 degrés de libertés, etc.). Khepera ne peut d'ailleurs pas être utilisé pour des applications qui nécessiteraient ce type de capteurs ou d'actionneurs. En résumé, avec Khepera on peut dégrossir le problème à un niveau légèrement supérieur que celui de la simulation, mais on ne peut pas passer au niveau des applications.

Pour s'approcher à des applications réelles, il faut passer à une taille qui s'adapte mieux au monde dans lequel le robot doit opérer. Pour cette raison, avec Edo Franzi et André Guignard, nous avons développé Koala, un robot de taille nettement supérieure (30-40 cm de diamètre) mais ayant des caractéristiques de base très similaires à celles de Khepera, notamment le même type de système moteur, le même type de capteurs et la même structure modulaire. Ce robot devrait permettre de transférer à un niveau plus applicatif les résultats obtenus sur Khepera. Ce transfert est facilité par une compatibilité logicielle et par les caractéristiques fonctionnelles similaires des deux robots.

Figure 20: Le robot Koala mesure environ 30 x 30 x 20 cm

3 L'INTERACTION AVEC L'ENVIRONNEMENT

Un robot mobile fonctionne dans son interaction avec l'environnement. Dans cette relation, les deux éléments, robot et environnement, sont aussi important l'un que l'autre, au point qu'on peut changer la tâche du robot en ne changeant que l'environnement, et en laissant le robot et sa programmation inchangés. C'est ce qu'on a réalisé dans les deux expériences présentées dans ce chapitre. Ces résultats ne veulent pas seulement montrer l'équivalence des deux éléments, robot et environnement, mais veulent aussi illustrer l'importance de la relation très complexe qui existe entre eux.

3.1 Introduction

L'interaction avec l'environnement est un des aspects les plus importants pour l'accomplissement d'un travail ou même pour la survie d'un système autonome dans le monde réel. Dans ce chapitre, l'importance de ce principe est illustré concrètement par deux expériences basées sur des données biologiques concernant le comportement des fourmi.

La fourmi est un des insectes les plus développés. Elle vit normalement dans une société très bien organisée et efficace. Dans le cadre de cette collectivité et seulement dans ce cadre, ces insectes sont capables d'optimiser des trajets de recherche de nourriture, de porter ensemble des objets de grandes dimensions, de construire des fourmillères etc. Malgré l'impression que l'on peut avoir en tant qu'observateurs du comportement de ces insectes, plusieurs scientifiques du domaine [Franceschini91] [Braitenberg84] [Deneubourg83] sont d'accord en affirmant que les mécanismes mis en jeu sont parfois étonnamment simples et efficaces. Un premier exemple de ceci est justement l'optimisation de trajectoire que les fourmi font entre une source de nourriture et la fourmillière. Lorsqu'une fourmi, pendant sa recherche, trouve une source de nourriture, elle prend ce qu'elle peut transporter et rentre au nid en laissant sur le sol une trace de phérormones. Une fois la fourmi arrivée au nid, cette trace servira de guide à d'autres fourmis pour retrouver la source de nourriture. Ces fourmis laisseront à leur tour une trace sur le sol. Dans le cas où plusieurs voies sont découvertes pour aller du nid à la nourriture, il n'y aura au début pas de trajet préférentiel. C'est lors de l'utilisation que le trajet plus court, sur lequel les fourmis retournent plus vite, sera légèrement plus marqué par les phérormones, invitant ainsi d'autres fourmis à emprunter ce chemin plutôt que l'autre. Progressivement et grâce à ces lois très simples des probabilités, le trajet le plus court sera donc privilégié.

Deneubourg et son équipe ont analysé en détail le comportement isolé et collectif des fourmis et d'autres espèces d'insectes collectifs. Deneubourg et Beckers ont essayé de modéliser le comportement des fourmis dans des situations bien précises afin de mieux comprendre le rôle des interactions entre individus et avec l'environnement pour l'accomplissement d'une tache globale. Ils ont vérifié leur modèles en simulation [Deneubourg91], et sur des robots mobiles [Beckers94] montrant la validité de leur théo-

rie. Les deux expériences décrites dans cette section reprennent ces travaux, illustrent le même principe sur des robots mobiles ayant une structure et un fonctionnement différent, et étendent l'analyse à une expérience nouvelle qui implique une coopération entre les robots. Une observation des caractéristiques de base de ces deux expériences permet d'illustrer l'importance de l'interaction entre système de contrôle, corps du robot, tâche et environnement. La prise en compte correcte de cette interaction est essentielle pour la bonne réussite d'un développement dans le monde technique en général, et de la robotique mobile autonome en particulier. Dans un monde aussi complexe que celui dans lequel un robot mobile agit, une modification très petite de cette interaction peut causer des changement radicaux dans la tâche et l'efficacité du système, comme le montrent bien les deux expériences de ce chapitre.

Enfin ces expériences sont un bon exemple de méthode synthétique, dans laquelle l'investigation biologique et technique se rencontrent sur le terrain de la robotique mobile autonome.

3.2 Le Travail Collectif

La première expérience est basée sur les travaux de Deneubourg qui portent sur le ramassage et triage d'objets dans une fourmillière. Deneubourg et Beckers ont modélisé, simulé et enfin implémenté sur des robots le comportement élémentaire des fourmis pour cette tâche, mettant en évidence le rôle de la stigmergie, ou "incitation par le résultat du travail" [Beckers94]. Les fourmis sont en effet en mesure de ramasser et entasser de la nourriture ou des fourmis mortes dans des endroits bien distincts et de façon apparemment coordonnée. Deneubourg et Beckers soutiennent que les mécanismes présents dans la fourmi pour effectuer cette tâche sont très simples: la fourmi ne fait que ramasser les objets qu'elle trouve et les déposer là ou elle retrouve d'autres objets du même type. Le fait d'ajouter encore un objet sur le tas ne va que renforcer la motivation des autres fourmis à mettre de nouveau ce type d'objets sur ce même tas. D'où l'"incitation par le résultat du travail". L'interaction avec l'environnement joue ainsi un rôle clé pour le comportement collectif. On parle d'ailleurs aussi de "communication par l'environnement", car l'entassement indique aux autres individus l'endroit où entasser.

L'expérience de vérification robotique porte seulement sur l'entassement et non sur le triage, ce qui est plus complexe à réaliser. Philippe Gaussier est le premier qui a réalisé une expérience avec un groupe de robots Khepera afin de valider la théorie et les simulations de Deneubourg [Gaussier94]. Beckers a répété et quantifié cette expérience avec des robot de taille moyenne (2 Kg) qui disposent, sur l'avant, d'une sorte de chasse-neige en forme de U leur permettant de pousser des objets (figure 21). Derrière le chasse-neige un interrupteur est déclenché dès que plus de trois objets sont poussés. Le robot bouge aléatoirement jusqu'à ce que l'interrupteur soit activé. A ce moment le robot recule, tourne et repart pour sa promenade aléatoire. Ce comportement a l'effet de regrouper les objets éparpillé sur le sol et les regrouper, selon Beckers, dans un seul et unique tas. Ce travail collectif est donc dirigé par l'état de l'environnement, qui est à la fois le résultat et la motivation du comportement. De plus l'environnement fait d'interface entre les différents robots présents.

Lors de l'analyse des résultats des expériences faites [Beckers94], Beckers va plus loin, affirmant que cette collaboration augmente l'efficacité des robots, exprimée sous la forme de travail total divisé par le nombre de robots.

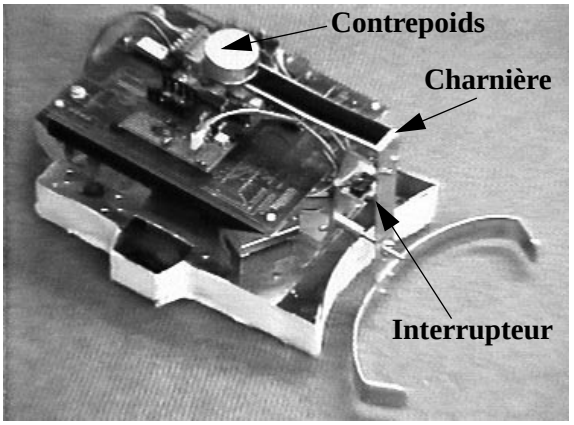


Figure 21: Le robot utilisé pour les expériences de Beckers et Deneubourg peut pousser des objets avec une sorte de chasse neige qu'il possède à l'avant. Dans sa position de repos, le bras est tenu éloigné du corps du robot par un contrepoids. Lorsque le frottement des objets sur le sol égale la force du contrepoids, le bras se rapproche du corps du robot et enclenche un interrupteur

3.2.1 Les données de l'expérience

Sur la base du premier essai fait par Gaussier [Gaussier94], l'expérience d'entassement a été reconçue, réalisée, quantifié, et les résultats de cette expérience analysés. Contrairement au premier essais de Gaussier, dans cette expérience Khepera a été équipé d'un bras pour saisir les objets (figure 22). Les capteurs utilisés pour détecter les objets sont les huit capteurs de proximité à lumière infrarouge (six à l'avant, deux à l'arrière), qui mesurent la lumière réfléchie par l'environnement à partir d'émission de lumière de la part du robot. Les objets que le robot ramasse sont des cylindres en bois d'environ 17 mm de diamètre et 25 mm de haut qu'il peut saisir et transporter avec son bras.

Figure 22: Robot Khepera dans la configuration utilisée pour les expériences collectives: la base est munie de 8 capteurs de proximité et la pince permet de saisir des objets (photo A. Herzog)

Le comportement implémenté sur Khepera est illustré dans la figure 23, algorithmiquement dans l'organigramme de la figure 25, et fonctionne de la façon suivante: Au démarrage le robot bouge aléatoirement (trajectoires avec une courbure aléatoire modifiée régulièrement) et est attiré par les obstacles qu'il perçoit avec ses 6 capteurs frontaux. Le comportement d'attirance vers les obstacles est réalisé par des connexions excitatrices et inhibitrices entre capteurs et moteurs comme l'illustre la figure 24a.

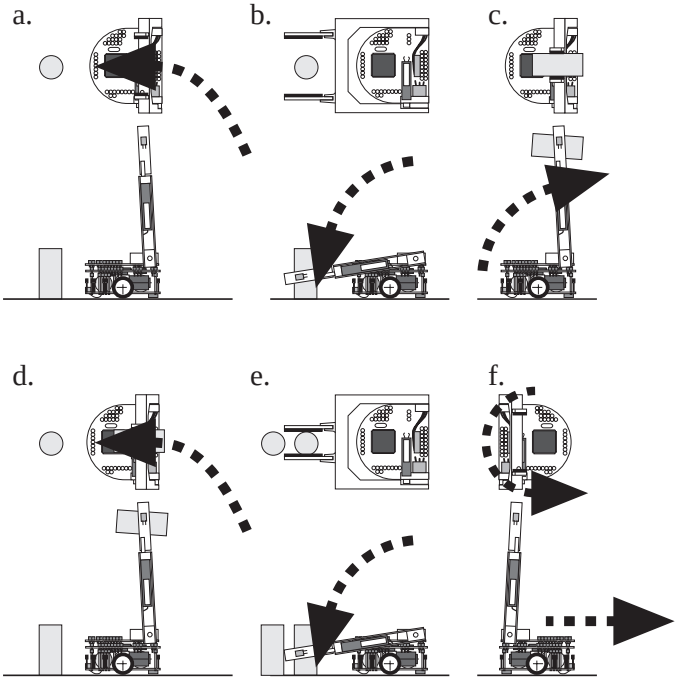


Figure 23: Séquence d'actions pour la créations de tas d'objets

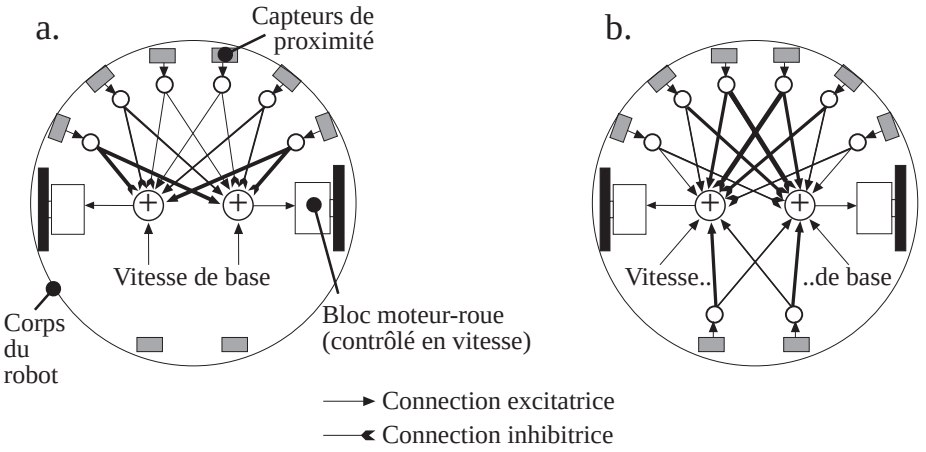


Figure 24: Structures de contrôle pour les comportements réflexes d'attirance (a.) et de répulsion (b.) par rapport à des obstacles. Ces structures sont réalisées par des connexions directes pondérées entre capteurs et effecteurs. L'épaisseur des traits des connexions indique leur importance. Les capteurs sont activés inversement proportionnellement au carré de la distance avec l'objet qui se trouve en face (voir aussi figure 15 à la page 40)

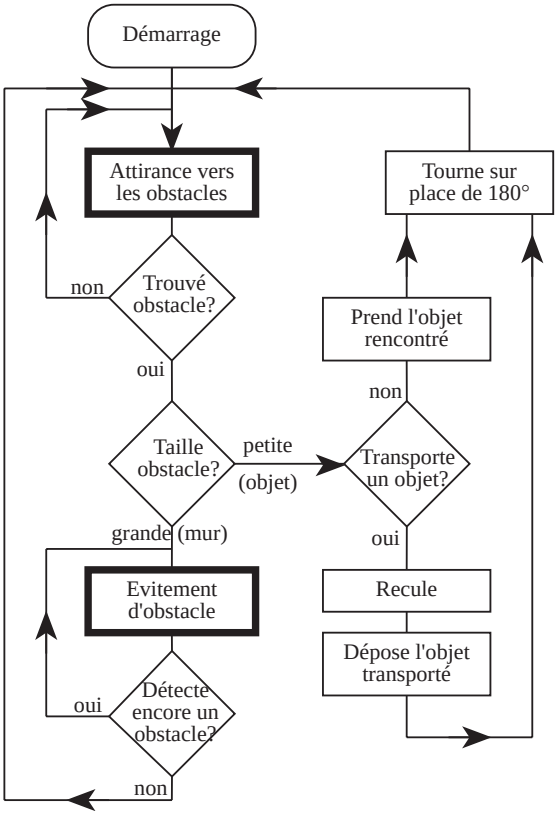


Figure 25: Organigramme de l'algorithme implémenté sur Khepera pour effectuer les deux expériences présentées dans cette section. Les opérations encadrées par un trait épais, sont réalisées par les réseaux de neurones de la figure 24

Lorsque un obstacle est détecté, le comportement d'attirance fait que le robot tourne et se dirige droit et centré sur l'obstacle. Une fois très proche (situation reconnue par le niveau des deux capteurs de proximité numéro 2 et 3 dans la figure 26a) le système de contrôle effectue une reconnaissance afin de vérifier si l'obstacle détecté est un objet ou un mur. Cette classification se fait par une vérification du niveau d'activation des deux capteurs avant latéraux (à droite et à gauche des deux capteurs centraux, numéro 1 et 4 dans la figure 26a). Si ce niveau est élevé, ceci signifie que l'objet rencontré est de grande dimension (figure 26c) et classifié comme étant un mur. Le système de contrôle enclenche alors un comportement d'évitement d'obstacles, réalisé aussi par des connexions excitatrices et inhibitrices entre capteurs et moteurs comme l'illustre la figure 24b. Ce comportement est maintenu jusqu'au moment où plus aucun obstacle n'est détecté au niveau des capteurs. Ensuite le comportement d'attirance est réactivé.

Si lors de la détection de l'obstacle le niveau des capteurs latéraux reste faible (en dessous d'un certains seuil, comme le montre la figure 26b), ceci signifie que l'objet est de faibles dimensions et est classifié comme étant un objet. Si le robot n'a rien dans sa pince, il saisi l'objet qu'il vient de trouver. Dans le cas qu'il a déjà un objet dans sa pince, il dépose l'objet transporté devant l'objet trouvé. Dans les deux cas, une fois la manipulation finie, le robot tourne sur lui-même de 180° et repart en activant le comportement d'attirance.

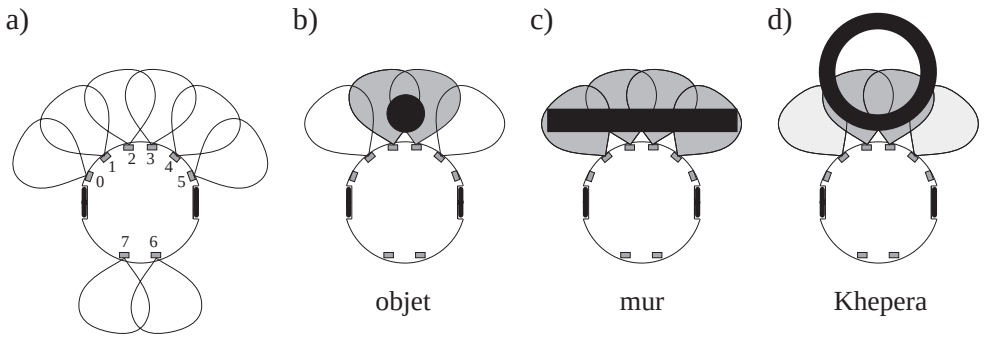


Figure 26: Illustration de la méthode de reconnaissance de la taille d'un obstacle basé sur les lobes de détection des 4 capteurs frontaux.

Pour des obstacles de taille moyenne (figure 26d), comme peuvent l'être les autres robots présents dans l'environnement, la reconnaissance peut porter à les classer dans l'une ou l'autre des catégories, selon les conditions de la détection.

Le robot programmé comme décrit ci dessus a été placé dans un environnement de 80 x 80 cm délimité par des murs et remplis de 20 petits cylindres en bois.

3.2.2 Résultats

Le résultat a bien été un ou plusieurs robots qui cherchent, saisissent, transportent et entassent des objets, avec un environnement résultant comme celui décrit dans la figure 27.

Figure 27: Situation typique avant et après l'expérience

La forme des tas crées est proche du linéaire, car le robot peut poser uniquement s'il détecte un objet isolé, et ceci arrive seulement aux bouts des tas, comme le montre la figure 28. Devant des larges tas le robot ne dépose pas car il reconnaît le tas comme étant un mur et l'évite.

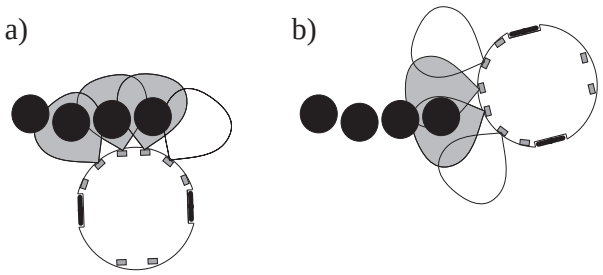


Figure 28: a) Détection de côté d'un tas d'objets reconnu comme un mur et évité. b) Détection d'un tas d'objets au bout, reconnu comme étant un objet et permettant la pose.

Le nombre de tas et leur taille ont été mesurés en fonction du temps et moyennés sur une série de trois expériences. Ensuite la même série d'expériences a été répétée avec 2, 3, 4 et 5 robots. Les courbes des résultats de ces essais sont illustrées en figure 29.

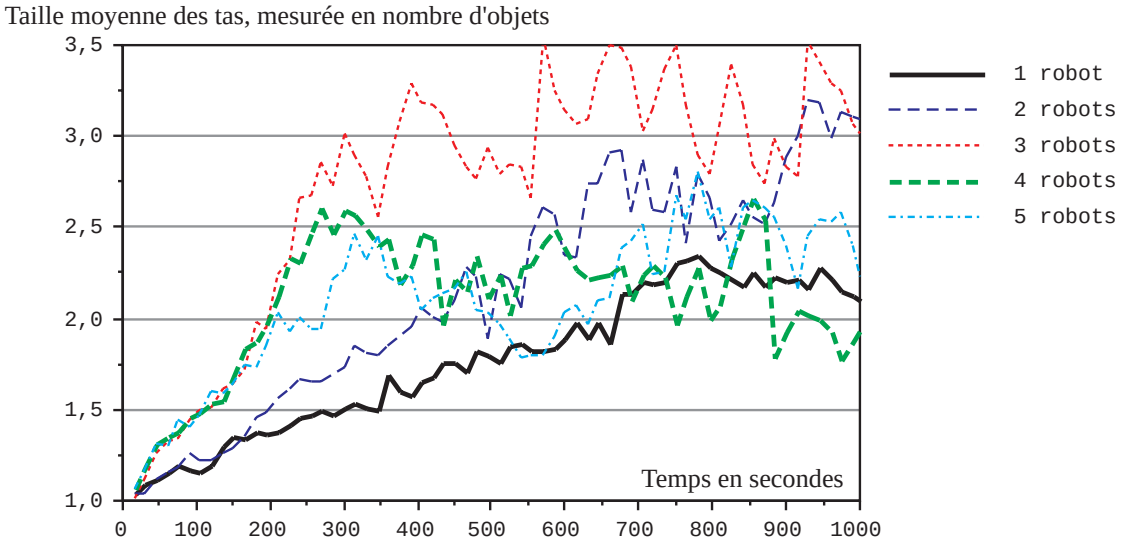


Figure 29: Quantification du travail d'entassement des robots, en fonction de leur nombre dans la zone de travail. Ces courbes sont le résultat du moyennage des mesures prises sur trois expériences

3.2.3 Analyse et discussion

Les résultats montrent d'une façon générale que les robots, au début et très rapidement, rassemblent les objets et forment des tas de plus en plus grands. Ensuite, après une période de forte croissance, la taille moyenne se stabilise entre les deux et les trois objets par tas. La vitesse à laquelle on atteint cette situation stable dépend du nombre de robot en jeu.

La raison de la formation de tas et de leur taille moyenne stabilisée entre 2 et 3 est à rechercher de nouveau dans l'interaction que le robot a avec son environnement, composé ici par des murs, des objets saisissables et des autres robots (à part pour l'expérience comportant seulement un robot).

3.2.3.1 Les interactions entre robot et murs

Les interactions avec les murs sont très répétitives et typées: Le robot est d'abord attiré vers le mur, en détecte la proximité, puis la taille et l'évite. Parfois il peut y avoir mauvaise détection et dans ce cas, si le robot transporte un objet, il peut y avoir création d'un tas d'un élément. Cet élément se trouvant proche d'un mur, il sera plus difficile à distinguer et à saisir. L'interaction avec les murs tend donc à faire baisser, de façon très faible, la taille moyenne des tas. Cette interaction étant faible, le rôle majeur est joué par les interactions avec les objets saisissables et avec les autres robots.

3.2.3.2 Les interactions entre robot et robot

L'interaction avec d'autres robots est assez complexe, car ils sont mobiles, réagissent aux perturbations et leur taille se situe juste à la limite des objets saisissables. Selon l'éclairage, le point de vue, les mouvements et la direction du robot rencontré, le robot à la recherche d'objets (appelé *chercheur*) peut reconnaître un robot qui se trouve sur son chemin (appelé *trouvé*) comme étant un objet ou un mur. Dans le cas où le robot trouvé est assimilé à un mur, l'évitement est enclenché et rien ne se passe. Si le robot trouvé est reconnu comme un objet, plusieurs cas de figure peuvent se présenter, selon le comportement du robot trouvé:

1. Si le robot trouvé se déplace environ dans la même direction que le robot chercheur, il peut y avoir poursuite sur une certaine longueur. Ce phénomène ne dure normalement pas longtemps car le robot trouvé, qui est souvent chercheur lui-même, détecte la présence du suiveur et réagit.
2. Si le robot trouvé est aussi chercheur lui-même, s'il détecte la présence du robot chercheur et qu'il le prend pour un mur, deux cas de figure se présentent: a) le robot chercheur ne transporte rien et il va essayer vainement de saisir l'autre robot (qui entre-temps se sera éloigné avec un comportement d'évitement). b) le robot chercheur transporte un objet et le dépose à l'endroit où il a rencontré le robot trouvé.
3. Si le robot trouvé est aussi chercheur lui-même, s'il détecte la présence du robot chercheur et qu'il le prend pour un objet, trois cas de figure se présentent: a) aucun des deux robot ne transporte d'objet: les deux robot vont essayer de se saisir mutuellement en se bloquant l'un l'autre. b) seulement un des deux robots transporte un objet: si la détection se fait simultanément sur les deux robots, l'un pose et l'autre saisi ce que le premier à posé. Il y a donc échange de l'objet. Si la détection n'est pas faite simultanément il y a simplement création d'un tas de un objet. c) les deux robots transportent un objet: les deux vont poser et former un tas de deux objets.
4. Si le robot trouvé est en train d'effectuer une reconnaissance, une prise ou une dépose (cas le plus fréquent), on retrouve les cas de figure a) et b) du point 2.

En résumé, ces interactions entre robots causent, le plus souvent des cas, une perte de temps, sans influencer directement la création ou la destruction de tas d'objets. Dans

les autres cas il y a principalement création d'un tas d'un seul objet, ce qui provoque une diminution de la taille moyenne. Seulement dans un cas assez rare l'interaction donne lieu à un tas de deux objets.

L'interaction entre robots la plus intéressante en soi est celle rencontrée dans le cas 3b et qui consiste dans l'échange d'objets. Gaussier, tout en utilisant une autre technique de saisie des objets, avait déjà remarqué ce comportement émergent. C'est en effet un comportement qui n'est pas programmé explicitement dans les règles de conduite du robot, tout comme le comportement principal d'entassement, et qui résulte essentiellement d'une fausse détection (deux robots qui se prennent pour des objets). Dans le cas de l'échange d'objets, l'effet du résultat par rapport à ce qui est programmé dans le robot est encore plus étonnant, étant donné l'impression que l'observateur peut avoir que l'échange est une vraie coordination, une concertation, une action qui semble basée sur de la communication. Cette observation illustre très bien l'affirmation de Brooks qui dit que l'"intelligence est dans les yeux de l'observateur" [Brooks91b] ainsi que les idées de Braitenberg sur l'observation des résultats comportementaux obtenus par des structures de contrôle simples [Braitenberg84].

3.2.3.3 Les interactions entre robot et objets

L'interaction entre le robot et les objets saisissables est essentiellement fonction de la configuration de ces derniers et de la situation du robot chercheur (transportant ou pas un objet). L'interaction se fait à deux reprises: lors de la pose (le robot transportant un objet) et de la saisie (le robot n'ayant pas d'objet dans sa pince). Lors de la pose, c'est le mécanisme de reconnaissance des objets qui règle le comportement du robot. Si le robot reconnaît l'obstacle rencontré comme étant un objet, il va poser l'objet qu'il transporte. Dans le cas contraire il va éviter l'obstacle rencontré et aucune pose n'aura lieu.

Cette interaction entre robot et environnement est caractérisée essentiellement par le type de détection qui dépend à son tour de la taille et la géométrie des tas d'objets. En supposant une forme de tas linéaire, ce qui est la tendance naturelle dictée par l'algorithme, on peut estimer géométriquement le type d'interaction robot-tas et en déduire des caractéristiques probabilistes.

La figure 30 illustre une estimation géométrique des zones de reconnaissance qui donnent lieu à la pose de l'objet. Le cas de la figure 30a représente la zone de reconnaissance d'un autre robot lorsqu'il est pris pour un objet. Ce cas correspondant à un tas de taille nulle et peut donner lieu à un tas d'un objet. Dans les cas b), c), d) et e) sont représentées les zones de reconnaissance et pose pour des tas de taille 1, 2, 3 et 4. Les tas de taille supérieure ont une zone de reconnaissance égale à celle de la figure 30e.

Un des paramètres intéressants de ces aires de reconnaissance est le périmètre de reconnaissance par rapport au périmètre total. En considérant que les objets utilisés ont un diamètre de 17 mm, les robots un diamètre de 55 mm et la distance de détection des objets est d'environ 25 mm, l'on obtient une estimation quantitative (tableau 2) des périmètres présentés dans la figure 30.

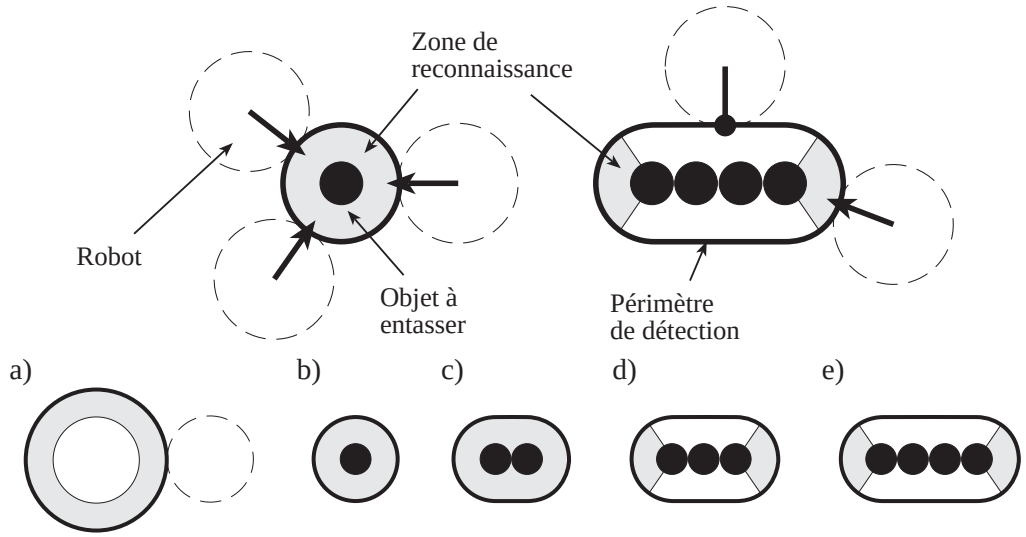


Figure 30: Périmètre de détection (en tait épais) et zone de reconnaissance avec classification dans la catégorie “objet” (en gris) en fonction de la taille du groupe d’objets. Pour des tas de taille supérieure à 4 objets, la zone de détection reste identique à celle du tas de 4 objets

Taille du tas en objets	0 (robot)	1	2	3	4	5	6	7	8	9
Périmètre de la zone de détection du tas [cm]	49.0	21.0	24.4	27.8	31.2	34.6	38.0	41.4	44.8	48.2
Partie du périmètre sur lequel se fait la reconnaissance du tas [cm]	49.0	21.0	24.4	13.0	13.0	13.0	13.0	13.0	13.0	13.0

Tableau 2: Périmètres de détection et de reconnaissance d’un tas en fonction de sa taille

On peut appliquer la même analyse géométrique au mécanisme de prise d’objets. Cette opération se compose d’une reconnaissance, dont les caractéristiques ont déjà été mises en évidence plus haut, suivie par une manipulation mécanique de l’objet. C’est cette dernière opération qui introduit le plus de contraintes, car plus limitée géométriquement que la détection à cause de la forme mécanique du préhenseur de la pince. La figure 31 illustre les zone d’accès de saisie, en fonction de la taille des tas dans lesquels l’objet à saisir se trouve. Pour des objets isolés, la zone de saisie est identique à la zone de reconnaissance. Pour des tas de plus grande taille, la zone d’accès est restreinte par la forme mécanique de la pince. En effet, si la pince n’arrive pas à se glisser sur les côtés de l’objet, celui-ci ne peut pas être saisi.

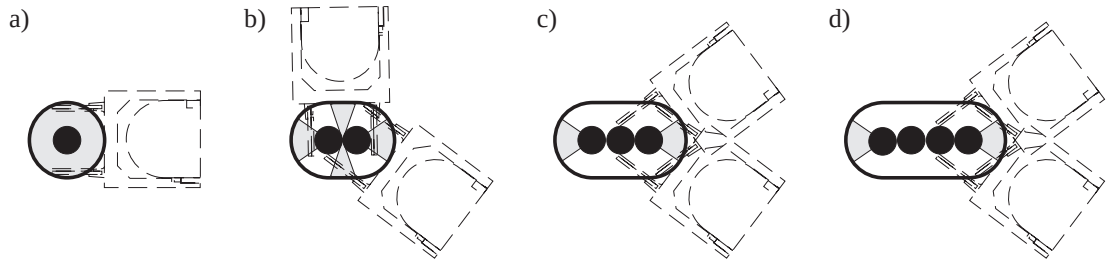


Figure 31: Zones d'accès pour la prise d'un objet pour différentes configurations de tas. Pour des tailles supérieures à 4 objets, la zone d'accès reste identique

Aussi dans ce cas il est intéressant de quantifier les périmètres d'accès destructif par rapport au périmètre total de détection du tas, en considérant comme base des paramètres identiques à ceux considérés pour le tableau 2. Ceci nous donnes les chiffres reportées dans le tableau 3.

Taille du tas en objets	0 (robot)	1	2	3	4	5	6	7	8	9
Périmètre de la zone de détection du tas [cm]	49.0	21.0	24.4	27.8	31.2	34.6	38.0	41.4	44.8	48.2
Périmètre de prise utile à la destruction du tas [cm]	0.0	21.0	12.5	8.1	8.1	8.1	8.1	8.1	8.1	8.1

Tableau 3: Périmètres de détection et de destruction d'un tas en fonction de sa taille

Une analyse des données contenues dans les tableaux 2 et 3 permet une première explication des fondements de l'interaction entre robot et objets. Le rapport entre le périmètre de détection pour la pose et le périmètre total (tableau 2) donnent une estimation de la probabilité de création d'un tas de taille supérieure lors de la détection d'un tas de taille donnée. De façon similaire le rapport entre périmètre de prise pour la destruction et périmètre total (tableau 3) donnent une estimation de la probabilité de destruction d'un tas de taille donnée. Ces rapports ainsi que la différence entre eux (construction moins destruction) sont représentés dans la figure 32. On peut ainsi facilement observer que la probabilité relative de construction d'un tas, pour des tailles supérieures à un objet, est toujours supérieure à la probabilité relative de destruction, ce qui permet de garantir une certaine stabilité des tas. La différence entre probabilité de construction et de destruction donne une indication quantitative de cette stabilité. Une comparaison de cette valeur sur l'ensemble des tailles de tas montre que les tas comportant deux ou trois objets sont les plus stables. Ce fait, associé à la faible stabilité des tas composé par un seul objet, explique bien la forte croissance initiale de la taille moyenne des tas reportée dans la figure 29 à la page 55. Cette forte croissance se ralentit à une taille de tas se situant entre 2 et 3, ce qui correspond aux configurations les plus stables.

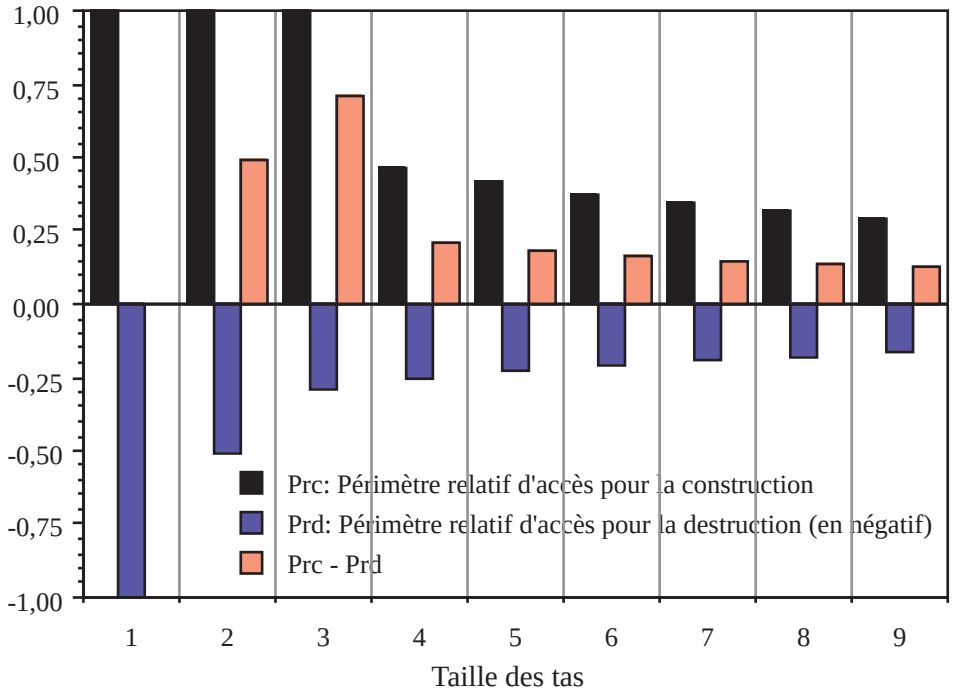


Figure 32: Illustration graphique du contenu des tableaux 2 et 3

Cette première analyse ne tient toutefois pas compte de la quantité de cas de figure que chaque taille de tas représente. Le cas de figure 30a, par exemple, à une plus grande zone d'accès que le cas de figure 30b, mais, lors du début de l'expérience par exemple, il y a beaucoup moins de robot que d'objets isolés. Si nous effectuons cette pondération en considérant un environnement contenant 5 robots et que l'on observe le nombre maximal de tas formé avec 20 objets on obtient le périmètre total de création ou destruction en fonction de la taille des tas. Ces données sont représentées en figure 33.

Dans le graphique de la figure 33 on peut observer encore plus clairement qu'il y a une très forte pression à détruire des tas formés par un seul objet, associée à une pression à créer des tas composés de deux et trois éléments. La baisse nette d'aire d'accès pour des tas plus gros stabilise la croissance de la taille des tas à une moyenne qui se situe bien entre 2 et 3 objets.

Le mécanisme de base qui permet d'obtenir le comportement d'entassement d'objets peut être déjà compris intuitivement par une analyse grossière des caractéristiques géométriques et probabilistes de l'interaction entre le robot et l'environnement, comme illustré dans cette section. Cette analyse reste toutefois statique, car elle ne tient pas compte de l'évolution dynamique de l'environnement, les déplacements des robots et la dynamique des tas dans leur taille et leur quantité.

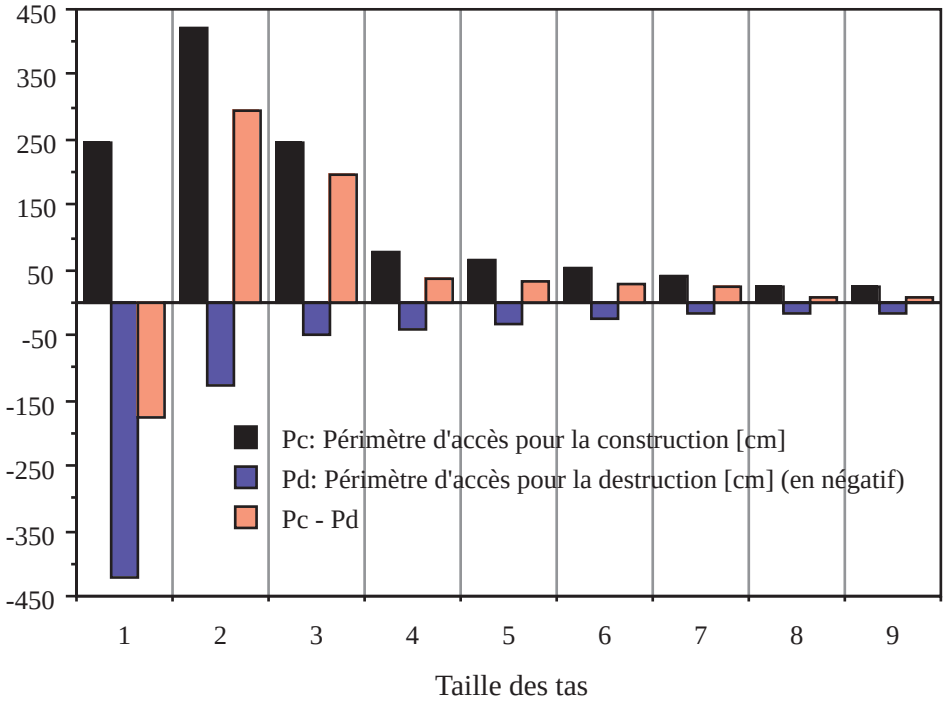


Figure 33: Périmètres totaux de création ou destruction de tas en tenant compte du nombre maximal de tas possibles et fonction de leur taille

3.2.4 La modélisation de l'expérience par une machine à états

3.2.4.1 La représentation sous forme de machine à états

La dynamique de cette expérience peut être formalisée sous forme d'une machine à états, dans laquelle on associe un nombre de tas à chaque taille possible. L'évolution de la machine à états est dictée par les probabilités qu'ont les objets de passer d'un tas à un autre. Ces probabilités sont décomposées en probabilités de quitter un tas (destruction) et probabilités d'être posé sur un autre tas existant (création d'un tas de taille supérieure).

Pour quatre tailles possibles de tas, une telle machine à états est représentée dans la figure 34. A chaque itération, le nombre de tas de chaque taille est mis à jour en fonction des transitions d'objets. Les passages d'objets d'un tas à l'autre sont faits proportionnellement aux probabilités associées, correspondantes à celles représentées dans la figure 32. Si par exemple il existe 10 tas de 3 objets et que la probabilité de destruction (D3) est de 0.3, 3 tas seront détruits pour former 3 objets en transport et trois tas supplémentaires de 2 objets. Pour garantir une certaine uniformité, tout passage est fait en nombre entiers, en respectant en tout moment le nombre d'objets qui se trouvent dans l'environnement.

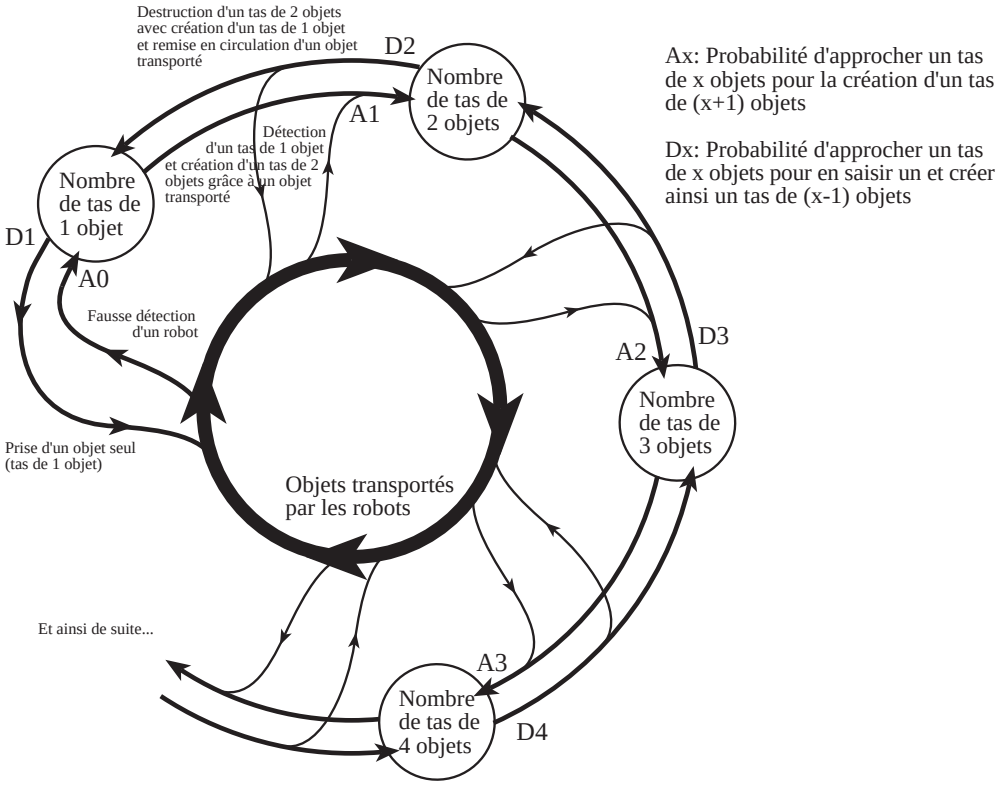


Figure 34: Graphe d'une machine à états représentant la dynamique du système de tas

On a fait évoluer ce modèle en considérant un environnement composé de 20 objets et d'un nombre de robots variant de 1 à 5. L'évolution de la taille moyenne des tas selon ce modèle et ces paramètres est donné en figure 35.

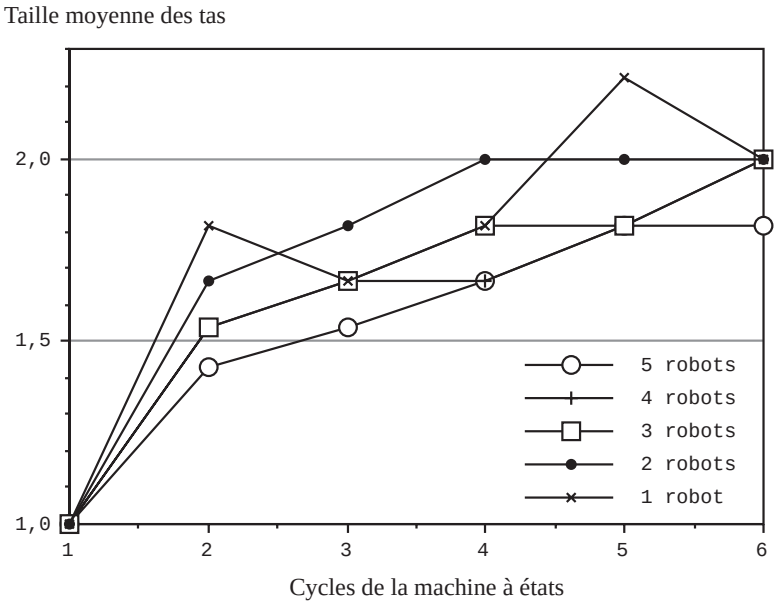


Figure 35: Evolution de la taille moyenne des tas en fonction du nombre de robots

3.2.4.2 Discussion

Cette représentation confirme que le jeu de probabilités données par les caractéristiques géométriques de l'interaction robot-environnement va dans la direction attendue, à savoir une forte croissance initiale qui se stabilise autour d'une taille moyenne se situant autour d'une valeur de 2.

Cette modélisation a toutefois des grosses lacunes au niveau de la quantification et de la prise en compte de l'aspect temporel. Le premier problème, lié à la quantification, est dû au fait que toute destruction ou création de tas doit prendre une valeur entière. Ceci signifie, en absence d'aspect temporel, qu'une probabilité basse peut ne pas permettre d'atteindre la valeur unitaire, ce qui annule l'effet de cette probabilité. L'effet de cette limitation est dans la moyenne de taille de tas relativement basse atteinte par ce modèle. Les probabilités de création de tas de plus grande taille étant basses, leur effet est annulé.

Le manque de l'aspect temporel en soi provoque aussi d'autres problèmes. Le nombre de robots influence ici seulement les probabilités de passage dues aux périmètres de prise et de pose, et non pas le travail réalisé par unité de temps. Dans le graphique de la figure 35 on remarque en effet qu'un robot atteint plus rapidement un grande taille moyenne que les cinq robots, ce qui est le contraire de ce qui a été mesuré lors de l'expérience (figure 29 à la page 55). Au niveau des périmètres en jeu, en effet, les cinq robots offrent beaucoup plus de possibilités de création de tas de un seul objet. Le fait que les cinq travaillent en même temps n'est par contre pas pris en considération.

3.2.5 La modélisation de l'expérience par une simulation

3.2.5.1 Le modèle de base de la simulation

Une simulation a donc été élaborée sur la base du fonctionnement probabiliste de la machine à états présentée plus haut, mais avec la prise en compte de l'aspect temporel et de la géométrie de l'environnement. L'essentiel de ce modèle temporel est présenté dans la figure 36: pour chaque itération de simulation et chaque robot on établit, sur la base de probabilités, si le robot détecte un obstacle dans l'environnement.

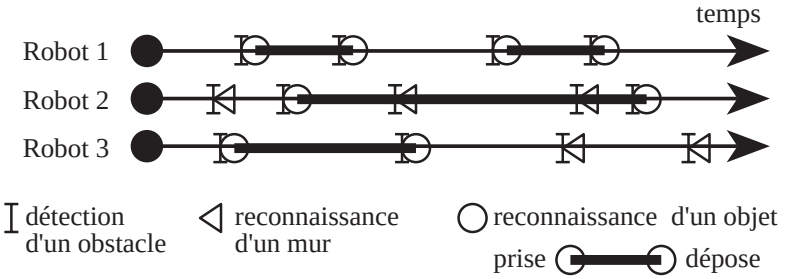


Figure 36: Modélisation temporelle des actions de plusieurs robots

Cette probabilité est établie selon une modélisation qui considère le robot comme étant un point qui se déplace aléatoirement dans l'environnement. Les obstacles sont représentés dans cet environnement par leur surfaces de détection (figure 37). Dans ce modèle on considère que le robot rencontre un obstacle lorsqu'il touche une des surfaces

de détection. La probabilité que ceci arrive correspond au rapport entre la somme des surfaces de détection de tous les objets présents et la surface totale de l'environnement. Le tableau 4 résume les données numériques nécessaires au calcul de cette probabilité.

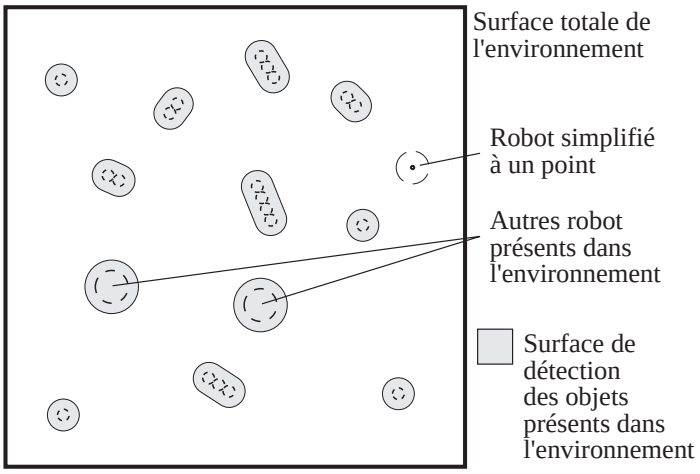


Figure 37: Surfaces prises en compte pour le calcul de la probabilité de détection d'un obstacle. Cette probabilité est le rapport entre la surface totale de détection des objets et la surface totale de l'environnement

Taille du tas en objets	0 (robot)	1	2	3	4	5	6	7	8	9
Surface de détection du tas [cm^2]	126.5	35.2	46.6	58.0	69.4	80.8	92.2	103.6	115.0	126.4

Tableau 4: Surfaces de détection d'un tas en fonction de sa taille

A chaque rencontre d'obstacle, un tirage aléatoire permet de déterminer avec quel tas on se trouve en contact et sur quelle partie du périmètre du tas la détection a été faite. Selon la partie du périmètre rencontrée, le robot peut reconnaître le tas comme étant un objet ou un mur. La figure 38 illustre graphiquement ce principe avec un exemple: On considère un environnement dans lequel se trouvent un tas de un objet, deux tas de deux objets et un tas de trois objets, ainsi qu'un robot. Si on aligne les périmètres de détection on obtient la colonne de gauche. Pour chaque périmètre de détection (en trait épais) une partie permet la prise d'un objet (en gris) et le reste ne donne pas lieu à une prise (en blanc). Un nombre aléatoire est choisi afin de déterminer quel tas et quelle partie du tas on détecte. Dans l'exemple de la figure 38, le nombre tiré est 0.48, qui sélectionne le premier tas de deux objets dans la partie de prise. La sélection d'une zone blanche (en tirant 0.1 par exemple) n'aurait donné lieu à aucune action. Une fois sélectionné le premier tas de deux pour la prise, un objet est chargé sur le robot et le tas de deux objets devient un tas de un objet. On trouve la nouvelle situation des périmètres sur la droite, avec en gris les zones de pose. Lors d'une itération successive, à condition que l'environnement n'aie pas changé et que le robot aie à nouveau détecté un objet selon le principe mentionné

plus haut, la situation est celle de droite. Un tirage aléatoire permet aussi de définir quel objet le robot a rencontré et si l'on peut poser l'objet. Dans le cas de la figure 38 la détection s'est faite sur le deuxième robot présent dans l'environnement, ce qui donne lieu à la création d'un tas d'un objet isolé.

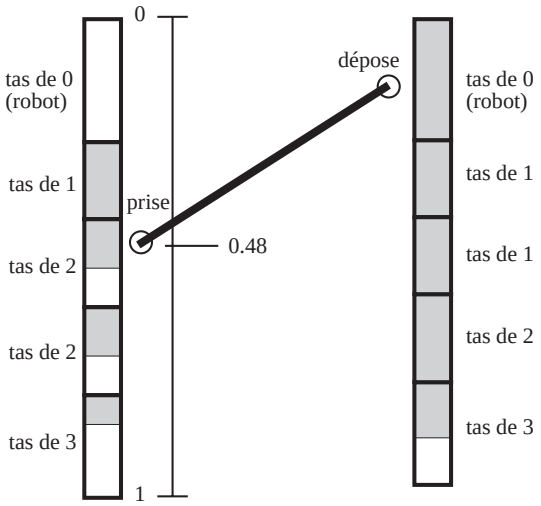


Figure 38: Sélection aléatoire des tas concernés en tenant compte des probabilités relatives de rencontre dues aux périmètres de détection, de prise et de pose

3.2.5.2 Les résultats de la simulation

Cette simulation est basée sur les mêmes paramètres que les essais précédents. Le nombre de robots a été fait varier de 1 à 5. Pour chaque cas, cinq simulations ont été faites et les résultats moyennés. La figure 39 illustre les courbes obtenues avec ce procédé.

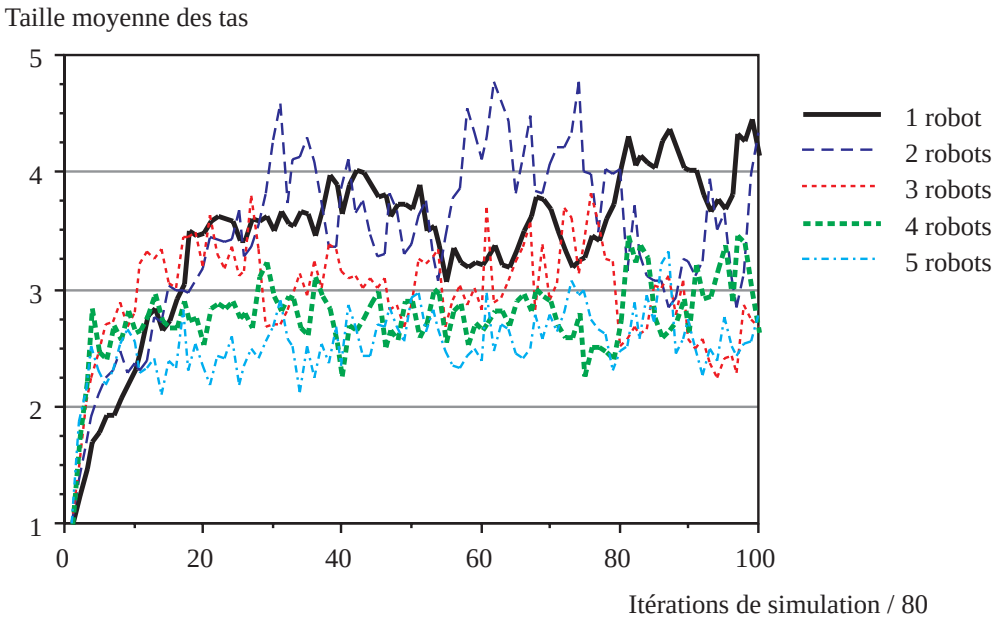


Figure 39: Taille moyenne des tas pour une simulation de l'expérience avec un à cinq robots et 20 objets. Ces valeurs sont le résultat d'un moyennage sur cinq simulations

3.2.5.3 Discussion

Les valeurs simulées illustrées dans la figure 39 et celles mesurées de la figure 29 (page 55) ont beaucoup de similitudes. L'allure générale confirme les hypothèses faites auparavant sur la croissance rapide initiale et la stabilisation à long terme. Dans cette dernière simulation la taille des tas se stabilise à des valeurs légèrement plus hautes que celles mesurées, mais ce point sera traité en détail plus loin. L'instabilité constatée sur les mesures se retrouve dans la simulation, et est due principalement au manque de stabilité qu'ont des tas de taille supérieure à 3. Aussi au niveau des performances entre les différentes tailles de groupes de robots, en simulation on retrouve, surtout dans la partie initiale, une différentiation similaire à celle rencontrée dans l'expérience réelle. Aussi sur ce point une discussion plus approfondie sera faite dans la section suivante.

Cette simulation permet donc de retrouver des comportements très similaires à celles mesurées, ce qui permet d'affirmer qu'une bonne partie des éléments essentiel au fonctionnement du système ont été pris en compte. Le comportement observé sur les robots est donc bien basé sur des lois statistiques fondées sur les caractéristiques géométriques de l'interaction entre le robot et son environnement. Cette interaction n'est pas seulement la base de fonctionnement, mais est de loin le point essentiel et complexe de l'expérience. L'affinement successif de la simulation fait dans cette section prouve cette complexité: la recherche des paramètres n'a pas porté sur le comportement, très simple, du robot, ni sur la forme de l'environnement en soi, mais sur les caractéristiques très fines et complexes de la relation robot-environnement. Les résultats sont d'ailleurs satisfaisants seulement grossièrement car, par exemple, la valeur asymptotique de la taille des tas trouvée en simulation ne correspond pas à celle mesurée. Ceci est du à la multiplicité des paramètres qui entrent en ligne de compte. Chaque paramètre peut être modifié et des nouveaux paramètres peuvent être ajoutés. Un élément qui n'a pas été pris en compte dans la simulation, par exemple, est le fait que les robots avaient de la peine à saisir des objets proches des murs. Or cette caractéristiques est difficile à modéliser, car n'est pas répétitive et est due à des fausses détections très difficilement caractérisables. Une possibilité de modélisation consiste à réduire la surface de travail en tenant compte d'une marge autour des murs. Pour tenir compte de cet effet au niveau de la prise, on peut corriger le périmètre de prise par un facteur correspondant au rapport entre la surface utile et la surface totale de l'environnement, la surface utile correspondant à la surface totale moins les marges. Avec ces nouveaux paramètres, les courbes de la figure 39 subissent des modifications importantes, comme le montre la figure 40 qui illustre la courbe de la taille moyenne des tas dans le cas d'un robot.

Ce changement de paramètre permettrait d'expliquer et corriger la valeur asymptotique des courbes de simulation, en atteignant un bon niveau de similitude avec les mesures de la figure 29, mais ceci n'est pas réaliste. A ce niveau, les estimations deviennent beaucoup trop grossières par rapport à la finesse d'ajustement des résultats, ce qui les rend peu réalistes et donc peu utiles. On atteint donc ici les limites de la simulation, principalement à cause de l'importance de beaucoup de comportements marginaux et difficilement caractérisables qui influencent le système global.

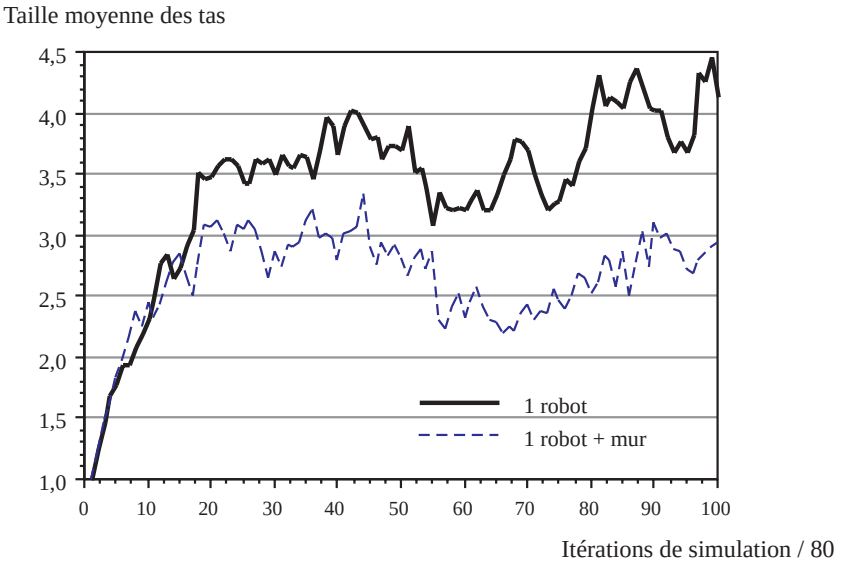


Figure 40: Effet de la prise en compte de l'influence des murs sur la courbe d'évolution de la taille des tas avec un seul robot

Malgré la relative simplicité de cette expérience, nécessaire pour pouvoir analyser les mécanismes et les résultats, ces observations permettent de mettre en évidence le rôle très important de l'interaction entre le robot et son environnement, ainsi que sa complexité. La simulation a permis de valider cette constatation, illustrant l'importance de l'interaction géométrique entre robot et environnement. La simulation a montré aussi ses limitations, même dans un cas aussi simple que celui-ci. Il est presque impossible de modéliser l'interaction entre deux robots en mouvement, par exemple. Or, ce genre de comportement joue un rôle important pour la valeur asymptotique du résultat.

L'importance de l'interaction entre robot et environnement sera encore plus mise en évidence et exploitée dans la deuxième expérience sur le comportement coopératif.

3.2.6 Le rôle de l'aspect collectif

L'aspect collectif, brièvement mentionné plus haut, est un autre point intéressant de l'expérience. Dans le graphique de la figure 29 on peut déjà observer que l'exécution du travail est accélérée par l'augmentation du nombre de robots. Gaussier [Gaussier94] et Beckers [Beckers94] affirment et essaient de prouver leur conviction que cette accélération n'existe pas seulement au niveau du groupe pris dans son ensemble, mais aussi au niveau de chaque robot pris individuellement. Ceci signifie, par exemple, que deux robots pourraient effectuer le travail plus rapidement que deux fois le travail d'un seul robot. En d'autres termes, la performance mesurée en quantité de travail par robot serait augmentée par la présence de plusieurs robots. Ce fait serait motivé par l'apparition d'interactions constructives entre robots, telles que la répartition des zones de travail, les collaborations directes (échanges d'objets), etc. Ce genre de phénomène apparaît par exemple lorsqu'un groupe de personnes, pour éteindre un incendie avec des sceaux d'eau, forme une chaîne dans laquelle personne ne se déplace et les sceaux sont passé

d'une personne à l'autre. Ceci garantit un apport continu et rapide d'eau qui est bien plus efficace que si chaque personne faisait le trajet.

On peut analyser cet aspect au niveau de notre expérience en retraçant le début du graphe de la figure 29 (page 55) avec, au lieu de l'axe du temps, un axe représentant le temps multiplié par le nombre de robot, comme le montre la figure 41.

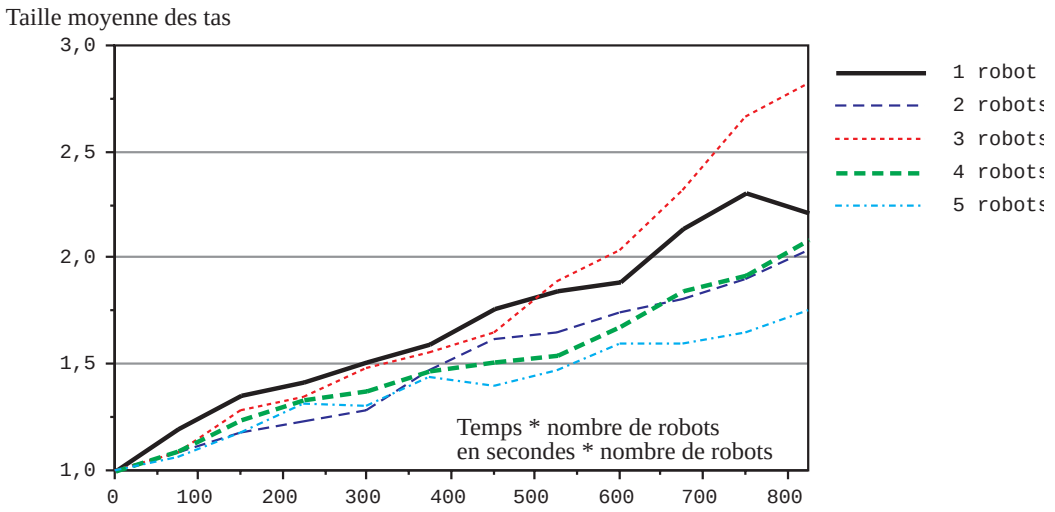


Figure 41: Quantification du travail d'entassement par robot individuel, en fonction du nombre de robots travaillant dans la zone de travail

3.2.6.1 Discussion

La représentation de la figure 41 montre bien que dans les premières 5 minutes, un robot travaillant seul est plus efficace que s'il travaillait en groupe. Cette caractéristique générale a été retrouvée dans la simulation faite plus haut (section 3.2.5), comme le montre la figure 42 (dans le format correspondant à celui de la figure 41).

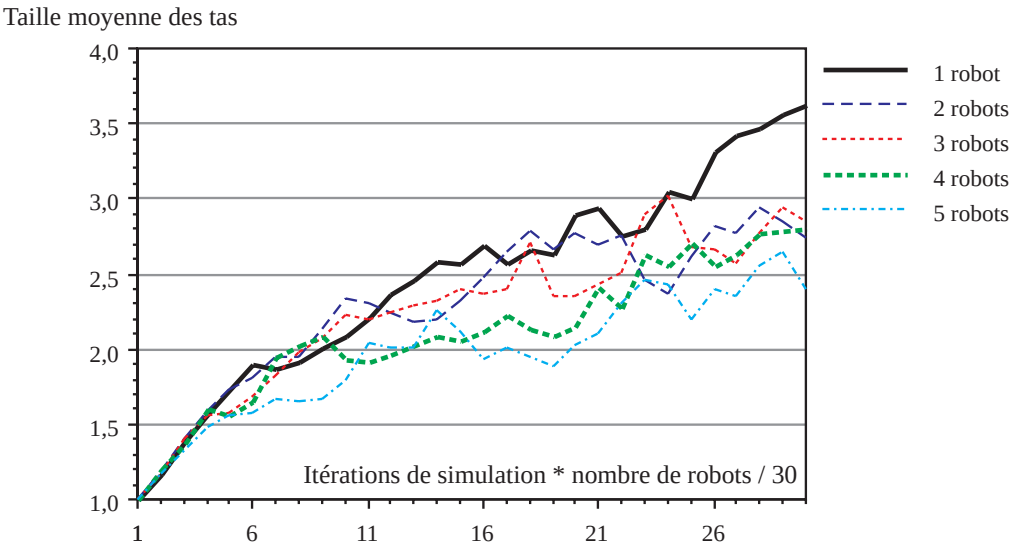


Figure 42: Simulation de la quantification du travail d'entassement par robot individuel, en fonction du nombre de robots travaillant dans la zone de travail

La diminution des performances dans le travail de groupe, dans ce cas bien particulier, peut être expliquée par deux raisons principales, toutes deux liées aux types d'interférences qui apparaissent entre les robots lors du travail collectif:

- Comme on l'a vu dans la liste des interférences possibles entre robots de la page 56, la plupart des interférences robot-robot ne sont pas constructives. L'interférence de type 1 n'a pas de répercussions, sinon que le robot chercheur est derrière un autre robot et ne peut donc pas trouver de nouveaux objets. L'interférence de type 2 cause de la perte de temps (cas a) ou la création d'un tas d'un objet (cas b). Les interférences 3a et 3b causent aussi de la perte de temps, alors que la 3c est la seule interférence constructive (formation d'un tas de deux objets) qui peut arriver. Pour l'interférence de type 4 on peut faire les mêmes observations que pour le type 2. Il est donc clair que les interactions robot-robot qui apparaissent lors du travail collectif dégradent l'efficacité de l'individu.
- Quantitativement au niveau des probabilités de pose et de prise, on remarque que le périmètre de création de tas de 1 objet, tel qu'il est calculé dans la figure 33 (périmètre de reconnaissance d'un robot multiplié par le nombre de robots) est proportionnel au nombre de robots. Ceci signifie qu'un robot seul a beaucoup moins de probabilités de créer des tas d'un seul objet que s'il y a d'autres robots dans l'environnement. La seule possibilité de former des tas d'un seul objet est de mal détecter un mur, ce qui arrive rarement. Un robot seul travaillera donc de façon bien plus efficace.

Dans ce cas précis, donc, l'aspect collectif n'apporte pas d'éléments constructifs dans le travail du groupe. La situation est donc similaire à celle d'un groupe de personnes qui veulent faire toutes le même travail au même endroit et qui finissent par se gêner plus que s'aider.

Dans l'expérience réelle et au delà des 500 secondes, c'est la configuration avec trois robots qui se montre plus efficace. Malgré que Beckers ait trouvé aussi des résultats analogues (meilleur résultat avec trois robots), ce résultat est très difficile à analyser, car sur le graphique de la figure 41, après 600 secondes*robots on se trouve à comparer la 600ème seconde de vie du robot seul avec la 300ème seconde de vie des robots à deux, avec la 200ème seconde de vie du triplet de robots etc. Or les conditions de l'environnement et des robots varient au cours de l'expérience. La première partie de vie est donc beaucoup plus représentative, car basée sur des conditions de base identiques. Ces conditions sont toutefois basées sur une distribution uniforme des objets. Lorsque les objets sont regroupés en tas et que la distance entre les objets augmente, l'augmentation du nombre de robot peut avoir des effets bénéfiques sur les performances, en plus de l'interaction 3c mentionnée plus haut: une bonne répartition géographique peut permettre une meilleure efficacité en minimisant les déplacements. Les interférences dues à d'autres robots peuvent sortir un robot d'une situation bloquée (robot enfermé derrière un tas d'objets, robot bloqué dans une situation symétrique, parfois insoluble pour le réseau de connexions implémenté dans cet algorithme). L'équilibre à long terme entre ces interférences bénéfiques et celles pénalisantes semble être dans ce cas très délicat, et pour cette raison difficile à évaluer.

Encore une fois, ces observations mettent en évidence l'importance que joue l'interaction entre robot et environnement. On comprend bien ici qu'une petite modification des caractéristiques de cette relation peut changer radicalement le comportement global du système. La section suivante s'occupe justement de ce cas et illustre ce type de phénomène par un exemple.

3.3 Le comportement coopératif

La première expérience décrite dans la section précédente peut être qualifiée de *collective*, car elle compte un nombre de robots supérieur à un, mais ne comporte pas de tâche nécessitant une réelle collaboration: chaque robot isolément, ayant un temps suffisant, peut résoudre toute la tâche. La collaboration n'est pas seulement superflue, mais dégrade les performances des robots par l'apparition d'interférences destructives. Dans la deuxième expérience, que l'on va décrire ci-dessous, nous avons voulu faire réaliser aux robots une tâche qui nécessite la coopération entre au moins deux robots. L'inspiration pour cette deuxième expérience vient aussi de la biologie des fourmis et des expériences faites à Bruxelles dans l'équipe de Deneubourg. Une des expériences qu'il a réalisé consiste à mettre, dans les environs d'une fourmillière, une série de bâtonnets en bois (allumettes) dans des trous faits dans le sol, de façon à que les bâtonnets sortent du sol sur environ le quart de leur longueur. Ces bâtons peuvent être utilisés par les fourmis comme matériel de construction. Or une fourmi seule n'arrive pas à sortir le bâton du trou. D'autre part cette situation particulière ne se rencontre que rarement dans la nature et ne peut donc pas être résolue avec des méthodes connues par les fourmis. La seule solution consiste en effet à que plusieurs fourmis se mettent ensemble pour extraire, de façon coordonnée, le bâton du trou. Pendant l'expérience avec les fourmis il a fallu peu de temps pour qu'elles sortent les bâtons des trous. Deneubourg affirme que ce résultat est le fruit d'un grand nombre de comportements simples qui font que de nombreuses fourmis tirent sur les bâtons de façon aléatoire et finissent par le sortir. Il n'y aurait donc ni communication explicite, ni collaboration concertée, mais simplement un large nombre d'individus qui, faisant de façon répété et aléatoire le geste très simple de tirer sur le bâton, finissent par le sortir.

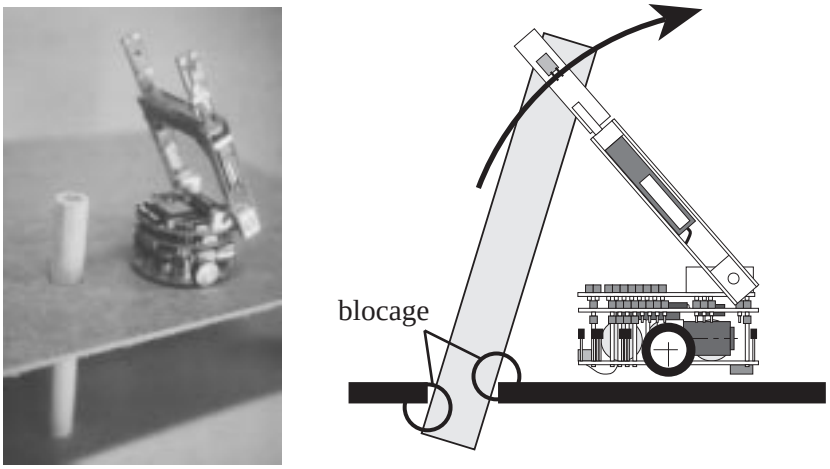
3.3.1 Les données de l'expérience

Dans cette deuxième expérience nous avons testé ces idées en prenant les robots programmés exactement comme pour la première expérience. Ces robots, comme les fourmis dans l'expérience décrite plus haut, ne sont donc pas censés connaître le problème mais sont programmés pour saisir des objets et les déplacer.

On a placé les robots dans un environnement de la même taille que celui de la première expérience, mais avec un fond surélevé et ayant des trous placés de façon distribuée sur la surface. Dans ces trous étaient placés des bâtons d'environ 150 mm de longueur, dépassant du sol d'environ 40 mm, comme le montre la figure 43a.

Comme dans le cas des fourmi, ici aussi un robot seul n'arrive pas à extraire un bâton du trou, essentiellement à cause de sa longueur et du mouvement circulaire de la

pince, qui fait que le bâton coince dans le trou (voir figure 43b). Il est donc nécessaire qu'au moins deux robots se mettent ensemble pour que l'extraction puisse avoir lieu.



a) b)

Figure 43: a) Robot et bâton dans l'environnement de l'expérience. b) Situation de blocage lorsqu'un robot essaie de sortir un bâton du trou

Afin que les robots ne restent pas bloqués en tirant sur le bâton, un délais de sept secondes a été introduit dans l'algorithme de contrôle, afin qu'après ce délais, le robot lâche sa prise, tourne de 180 degrés et continue sa recherche. Lorsque deux robots ont sorti un bâton du trou, celui-ci a été enlevé de la pince des robots et remis dans le trou, essentiellement afin d'éviter que les robots soient bloqués sur le trou.

Cette expérience a consisté à mettre 5 robots dans un environnement ayant de 2 à 6 bâtons. Chaque expérience a duré environ 20 minutes, correspondant à la durée des accumulateurs sous la charge importante de la pince.

3.3.2 Résultats

La quantification de cette expérience a porté sur les essais de prise, suivis par une réussite (sortie du bâton) ou pas. La plupart du temps les robots se déplacent aléatoirement, trouvent un bâton (ou un autre robot), essaient de le saisir et après sept secondes lâchent la prise et repartent. Seulement très rarement deux robots se succèdent pour tirer sur le bâton dans la bonne séquence temporelle qui leur permet de sortir le bâton du trou. Comme le montre la figure 44, il faut d'abord qu'un robot soulève le bâton au maximum qu'il peut. Si exactement à ce moment un autre robot, étant arrivé du bon côté et ayant reconnu le bâton, le saisis et commence à tirer, le bâton sera extrait dès que le premier robot lâchera sa prise. Cette séquence temporelle est due au pur hasard, et ne comporte rien d'autre que les mécanismes décrits plus haut.

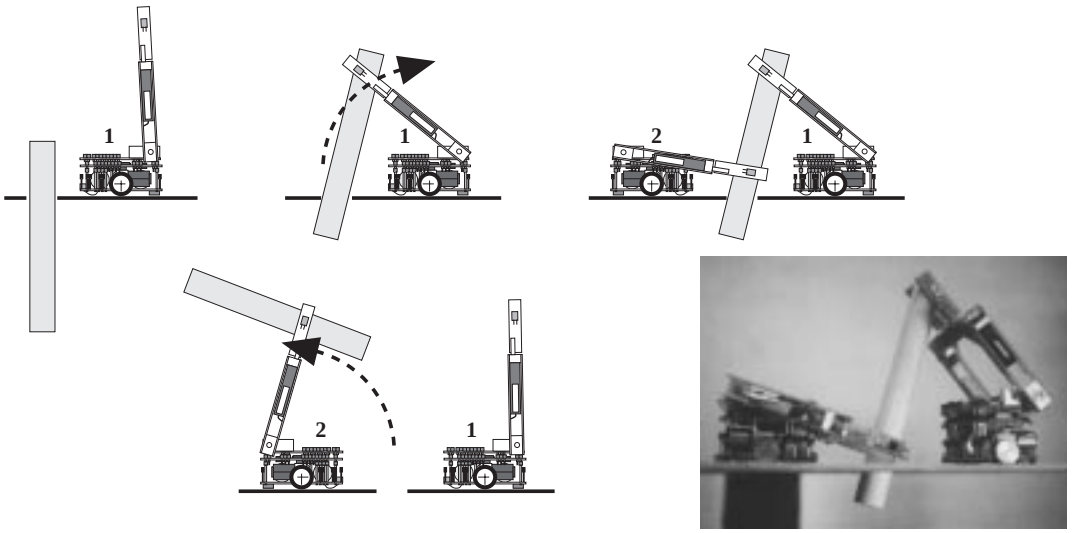


Figure 44: Séquence nécessaire pour sortir un bâton d'un trou

Le résultat principal de cette expérience est que les robots arrivent à extraire les bâtons des trous. Du fait du faible nombre de robot et du grand nombre d'interférences, l'extraction du bâton arrive très rarement. Dans les expériences faites, l'extraction est arrivée 6 fois pour une durée d'expérimentation d'environ 110 minutes.

Si on observe les très nombreux essais de saisie, on peut remarquer qu'il y en a deux types différents: les essais faits sur un bâton, et les essais faits sur un autre robot causés par une fausse détection. Le graphe de la figure 45 présente la quantité de ces événements prise sur une durée normalisée de 20 minutes. Il illustre la quantité de détections et prises de bâtons, ainsi que les interférences dues aux fausses détections d'autres robots, pris pour des bâtons.

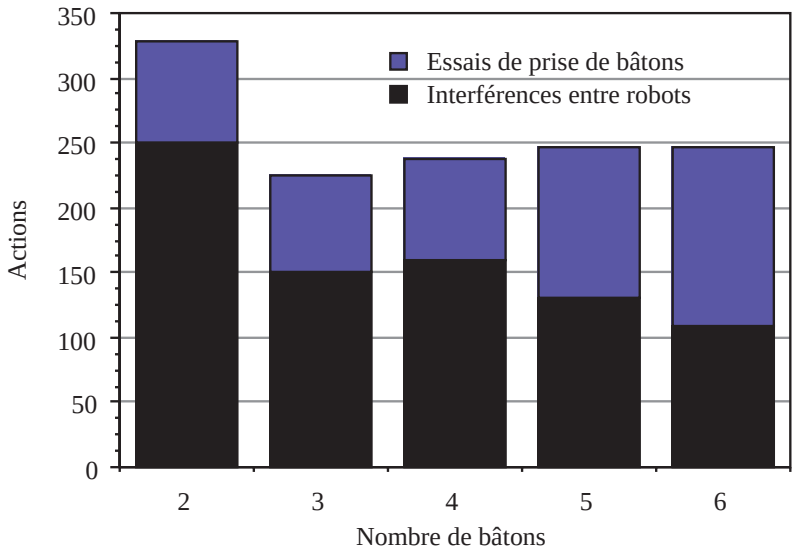


Figure 45: Quantification des essais de prise de bâtons et d'interférences en fonction du nombre de bâtons présents dans l'environnement

3.3.3 Discussion

Au niveau général comme au niveau du détail, on retrouve l'importance fondamentale de l'interaction robot-environnement.

En ce qui concerne les résultats détaillés des essais de prise, le rapport entre robots et bâtons présents dans l'environnement joue un rôle important. Cette importance est due au même phénomène d'interaction rencontré dans la première expérience, à savoir la fausse détection d'autres robots. Cette interférence, associée à la présence relative d'autres robots par rapport au nombre de bâtons, change fortement l'efficacité du système, ce qui se remarque bien dans le graphique de la figure 45: malgré un nombre d'actions totales qui reste presque constant, mis à part pour le cas extrême correspondant à deux bâtons, le rapport entre les interférences et les essais de prise augmente lorsque le nombre de bâtons diminue. Ce rapport dépend en effet fortement du rapport entre le nombre de robots et celui de bâtons présents, ce qui détermine la probabilité de rencontre d'un bâton par rapport à un autre robot. Cette relation est proche d'être linéaire, comme le montre la figure 46. Naturellement d'autres interactions viennent se mêler à celle-ci, comme le chevauchement d'interférences et prises, les détections "correctes" d'autres robots considérés comme des obstacles, les blocages mutuels des robots, etc. On peut facilement imaginer la complexité de l'interaction entre tous ces éléments, qui rend le phénomène difficilement analysable.

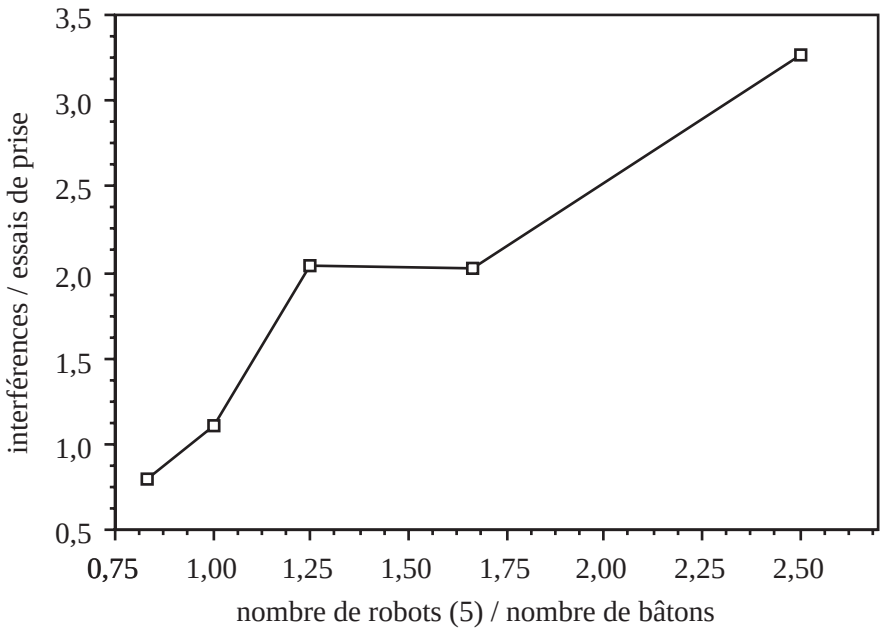


Figure 46: Relation qui lie le comportement (prises et interférences) avec le nombre de robots et bâtons présents dans l'environnement

Le nombre de robots était malheureusement limité à 5 pour des raisons de disponibilités de matériel, ce qui est loin de simuler des conditions d'une fourmillière. Nous avons toutefois retrouvé l'effet de probabilité associé au hasard que Deneubourg trouve dans sa fourmillière et qui permet aux Khepera d'effectuer la tâche d'extraction des

bâtons. Cette expérience semble donc confirmer les théories de Deneubourg.

Au niveau plus général de la tâche exécutée, il est très intéressant de voir comment des robots programmés d'une façon déterminée et identique peuvent exécuter deux tâches fondamentalement différentes dans deux environnements qui ne sont que légèrement différents. C'est en observant ce fait que l'on comprend mieux le rôle essentiel de l'interaction robot-environnement. Dans les deux expériences cette interaction permet de changer la fonctionnalité et l'efficacité des robots aux yeux de l'observateur.

En observant la grande différence de résultat de ces deux expériences qui utilisent les mêmes robots programmés de la même façon, on voit aussi l'importance de l'interprétation de l'observateur. Dans les deux expériences l'observateur donne une interprétation complètement différente (coopération ou pas) pour un comportement de robot identique.

3.4 Conclusion

Ces deux expériences, malgré leur simplicité, nous illustrent plusieurs aspects très importants:

- L'interaction du robot avec l'environnement joue un rôle fondamental pour la réalisation d'une tâche donnée: la forme et le nombre des objets, le nombre et la morphologie des robots, les caractéristiques des capteurs et des actionneurs conditionnent fondamentalement tout le mécanisme de fonctionnement. Une légère modification de ces conditions apporte des changements très importants de l'efficacité jusqu'à changer fondamentalement le rôle même du robot dans l'environnement. L'interaction entre robot et environnement joue donc un rôle tout aussi important que l'algorithme de contrôle qui gouverne le robot. Ceci est très clairement démontré par les deux expériences décrites dans ce chapitre. Le changement radical de mode de fonctionnement est dicté par un changement de l'interaction robot-environnement, et non pas par un changement d'algorithme, ce qui aurait pu aussi provoquer un tel effet.
- En regardant ces mêmes expériences et ce même aspect d'interaction sous un autre point de vue, on peut observer qu'une bonne exploitation des caractéristiques de l'interaction entre le robot et son environnement peut permettre de réaliser des comportements très intéressants avec des structures très simples par rapport à celles que l'on pourrait considérer à priori. Ceci nécessiterait l'exploitation des mécanismes d'interaction en tant que partie intégrante de la méthode de conception du système. Pour cette raison et pour l'importance de cet aspect, il est essentiel de prendre en compte les caractéristiques de l'environnement, du robot et de leur interaction, comme par exemple les probabilités de création et destruction de tas de la première expérience, afin de réaliser un système efficace. Cette exploitation, une fois mise en oeuvre, doit permettre de simplifier la conception. Ce type d'approche est opposée aux méthodes de conception classiques, qui considèrent l'interaction comme faisant partie du problème à

modéliser et à résoudre, comme le robot et sa tâche. Or l'interaction robot-environnement, comme on l'a vu dans ces expériences, est très complexe et souvent hors de portée de nos techniques d'analyse et de conception. Elle est donc plus souvent simplifiée et minimisée qu'exploitée dans les méthodes classiques basées sur une modélisation du système.

- Un autre point important, lié au précédent mais qui forme un point en soi, est le fait qu'une structure de contrôle simple peut donner lieu à un comportement compliqué. Ce concept est bien illustré par les deux expériences de ce chapitre, et doit être pris en considération dans les méthodes de conception. La réalisation de tâches compliquées doit se faire en exploitant intensément l'interaction robot-environnement, comme illustré au point précédent.
- L'association de comportements simples et la bonne exploitation de l'interaction robot-environnement donne lieu à ce que l'on appelle des *comportements émergents*. Souvent ce concept d'émergence a des connotations presque magiques: on programme un comportement et on en voit émerger une autre. Le concept d'émergence est aussi souvent associé au concept d'intelligence. "Si le robot fait une chose qui n'a pas été prévue, c'est qu'il dispose d'une certaine intelligence". C'est par exemple le cas de l'échange d'objets entre deux robots qui apparaît dans la première expérience. Ce comportement, vu de l'extérieur, est un nouveau comportement qui n'a pas été programmé et qui *émerge* pendant l'expérimentation. De même pour l'extraction des bâtons dans la deuxième expérience. Malgré cette connotation très peu scientifique et trop souvent utilisée, ce terme désigne très bien les phénomènes qui ont des caractéristiques bien déterminées pour l'observateur mais qui résultent de la combinaison de conditions que l'observateur ne relie pas directement entre elles et avec le résultat. Les mécanismes de prise et pose d'objets dans la première expérience donnent lieu à l'échange d'objets, que l'observateur voit comme étant un comportement en soi d'une complexité supérieure à ce qui est programmé dans les robots. Deux choses sont donc à retenir en ce qui concerne l'émergence: premièrement il s'agit d'un phénomène tout à fait logique et intéressant, car il permet d'exploiter la combinaison de conditions et de comportements. Deuxièmement, le phénomène d'émergence, d'un comportement par exemple, a lieu principalement dans l'esprit de l'observateur. Ici encore on retrouve le lien entre *émergence* et *intelligence* car pour ce dernier concepts aussi, Brooks affirme qu'il est essentiellement dans les yeux de l'observateur [Brooks91b].
- Si l'intérêt pour les phénomènes émergents est clair, et qu'il est clair aussi que ces phénomènes sont le résultat par excellence d'une approche bottom-up, il reste un grand problème de fond quant à la conception d'un système réel en utilisant cette approche. Dans les expérience décrites ci dessus, les robots, l'environnement et les tâches observées ont été obtenus par copie d'un exemple qui se trouve dans la nature. Dans le développement d'un système réel, comme pourrait être un robot de nettoyage, par

exemple, le but est donné à l'avance et une approche purement émergente ne peut pas être appliquée aveuglement. Il faut en effet diriger l'émergence, ce qui introduit des éléments de l'approche top-down. Il est donc nécessaire, afin d'obtenir des résultats utilisables, de se situer entre les deux approches.

- Le parallélisme mis en oeuvre dans ces deux expérience est un élément intéressant, même si l'efficacité individuelle des robots n'est pas forcément accrue. L'efficacité globale absolue est en effet accrue et ceci est déjà un avantage. Un autre avantage est par exemple la robustesse aux pannes. Dans les expériences il est arrivé qu'un robot soit coincé dans une situation imprévue ou difficile à résoudre (robot bloqué derrière un mur d'objets, par exemple). Ceci n'a pas bloqué le fonctionnement du système, car les autres robot ont continué à fonctionner. Bien au contraire, plusieurs fois, les actions réalisées par les autres robots ont débloqué le robot en difficulté, en le ramenant à des situations connues. Ceci est un point positif de l'interaction entre robots qui ne doit pas être sous-estimé.
- Le phénomène au niveau comportemental de l'interaction entre des agents simples et leur environnement pour atteindre des résultats qui à un niveau plus global semblent complexes, peut être transposé à d'autres domaines, comme par exemple celui des réseaux de neurones. Aussi dans ce cas on se trouve face à des éléments simples qui intéragissent et donnent lieu à des résultats complexes. Les méthodes qui s'appliquent à un domaine peuvent probablement aussi être appliquées à l'autre. Ce sujet ne sera pas approfondi dans le cadre de ce travail, mais est une des motivation de sa suite.
- Malgré le développement préalable d'outils de mesure et d'analyse du comportement (voir aussi la section 2.3 *L'outil Khepera*), ces outils se sont révélés peu adaptés aux besoins qui apparaissent lors d'expériences qui impliquent un nombre de robots supérieur à un. Pour pouvoir continuer les investigations dans le domaine du collectif il faudra donc prendre soin à combler cette lacune.

En résumé l'interaction entre système de contrôle, robot et environnement, la simplicité des approches et l'inspiration biologique semblent être des points très importants qui permettent, si considérés attentivement, d'atteindre des résultats très intéressants. Leur application est toutefois très délicate, comme le montre l'échec de l'approche de l'intelligence artificielle classique ou de certain réseaux de neurones, qui se veulent inspirés de la nature. Un effort tout particulier doit donc être fait pour comprendre les phénomènes en jeu et développer des méthodologies de conception qui permettent d'utiliser ces principes en association avec notre technologie. Finalement la modélisation reste très limitée par la complexité des phénomènes en jeu, même pour des expériences comme celles-ci qui paraissent simples.

4 L'APPRENTISSAGE

L'expérience illustrée dans la section précédente démontre que le comportement d'un robot mobile, s'il veut être efficace, doit être bien adapté à l'environnement dans lequel le robot agit. Dans les exemples du chapitre précédent, cette adaptation du robot à l'environnement a été finement ajustée, par le concepteur, sur l'exemple d'un modèle de la nature, afin d'atteindre les résultats voulus. Ce type de conception est possible dans des environnements et pour des tâches très simples ou en acceptant des tolérances de comportement très grandes. Avec l'accroissement de la complexité de l'environnement ou du comportement, la conception classique basée sur une modélisation du robot et de son environnement devient vite infaisable. Afin d'éviter la modélisation réalisée par le concepteur, il faut permettre au robot de construire ses connaissances et s'adapter sur la base de son interaction avec l'environnement.

Afin de donner cette propriété au système de contrôle, le premier pas consiste à le rendre adaptatif, en lui faisant apprendre certaines caractéristiques directement du monde réel. L'apprentissage est un des mécanismes les plus utilisés à cet effet, surtout dans le domaine des réseaux de neurones. Le présent chapitre a pour but de présenter l'approche neuronale, d'introduire l'apprentissage et d'en vérifier les possibilités. Dans ce but, nous analyserons une expérience utilisant cette technique dans un cas pratique de contrôle d'un robot mobile.

4.1 Introduction aux réseaux de neurones biologiques

Le système de contrôle des animaux est essentiellement basé sur le système nerveux. La brique de base du système nerveux animal est un type de cellule particulier, appelé neurone (figure 47). Organisé en réseau, ce type de cellule effectue la transmission des signaux entre capteurs et effecteurs ainsi que le traitement du signal transmis. Le traitement élémentaire du signal est fait au niveau du neurone et au niveau de ses connexions avec les autres neurones.

Le neurone est composé d'un corps principal, dans lequel se trouve le noyau, complété par des ramifications ayant pour but de recueillir (les dendrites) et retransmettre plus loin (les axones) le signal. A l'extrémité de l'axone se trouvent les synapses, qui, connectées aux dendrites d'autres neurones, s'occupent de la transmission du signal (figure 47).

La transmission du signal se fait par des impulsions électriques, appelées les potentiels d'action, dont l'amplitude est d'environ 100mV. Ces potentiels d'action sont générés par un pompage d'ions qui a lieu au niveau de la membrane de la cellule. Lorsqu'un neurone est excité, un potentiel d'action est généré à la racine de l'axone et se propage le long de l'axone, à une vitesse d'environ 100m/s. La durée de l'impulsion à un point donné est de l'ordre de la milliseconde. A son arrivée à la synapse, le potentiel d'action est converti en signaux chimiques, à savoir les neuromédiateurs. Le neurone qui émet ce signal est appelé neurone présynaptique. Les neuromédiateurs se diffusent dans la

synapse et se fixent à des récepteurs spécifiques qui se trouvent sur les dendrites du neurone récepteur, appelé neurone postsynaptique. Cette fixation provoque un changement de la perméabilité de la membrane du neurone postsynaptique. Chacune des synapses ne produit qu'un effet faible. C'est l'intégration continue du neurone, faite de façon non-linéaire, qui engendre un potentiel d'action.

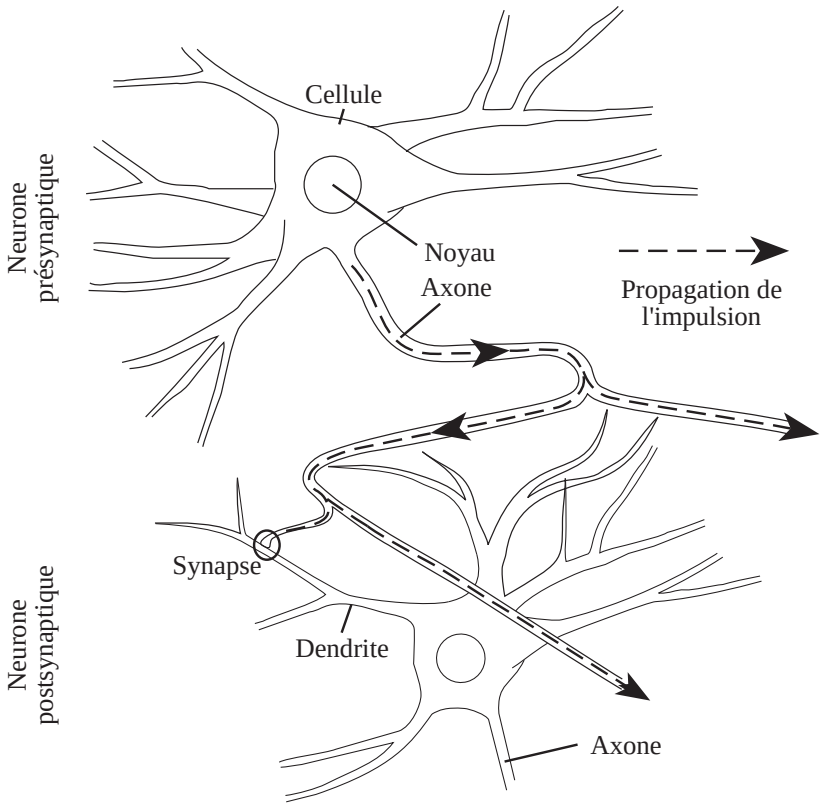


Figure 47: Structure d'un neurone biologique

Le système nerveux est un système adaptatif, capable d'apprentissage. Cette faculté d'adaptation a ses racines dans la plasticité du neurone. Les signaux transmis ne codent pas uniquement de l'information, mais modifient la structure et le métabolisme du neurone. Un lieu privilégié de cette évolution semble être la synapse. L'efficacité synaptique évolue constamment grâce, par exemple, à des variations des quantités de neuromédiateurs libérés en fonction du potentiel d'action ou du nombre de récepteurs fonctionnels. A un niveau structurel, l'activité neuronale peut modifier le nombre et la localisation des synapses, soit en modifiant les terminaisons de l'axone, soit en remodelant les arborescences dendritiques.

Le mécanisme de base décrit ici est un modèle général, mais les neurones ne sont pas tous identiques: dans le système nerveux, il existe une grande variété de neurones avec des caractéristiques physiques et fonctionnelles différentes.

Finalement ces neurones, comme nous avons pu le voir à propos de leur structure, sont organisés en réseaux. Le réseau global est normalement subdivisé en zones, dans

lesquelles les neurones ont une structure, des formes et un rôle fonctionnel spécifiques. Les interconnections sont très denses, avec des milliers de synapses par neurone.

4.2 Les réseaux de neurones artificiels

Le neurone artificiel et les réseaux de neurones artificiels (RNA) essaient de copier la version réelle. Cette copie peut être faite à différents niveaux, suivant le but visé. Les biologistes, qui veulent utiliser les RNA pour comprendre leur modèle biologique, essaient de s'approcher le plus possible de la réalité biologique en copiant la structure des réseaux, la réponse du neurone ou d'un groupe de neurones ou la fonctionnalité d'une zone [Franceschini92]. Les ingénieurs, intéressés par de nouveaux résultats dans le domaine du traitement de l'information, dévient très facilement des modèles biologiques pour différentes raisons: D'une part, le modèle biologique est compliqué et, d'autre part, il est adapté à des problèmes différents de ceux rencontrés dans le monde technique. Enfin ce n'est pas le modèle biologique qui motive les ingénieurs, mais les résultats que l'on peut obtenir. Pour cette raison, très souvent, l'inspiration biologique est remplacée par les méthodes de l'ingénieur qui permettent de maîtriser le problème et de le résoudre de façon méthodologique.

Dans le cadre de ce travail, nous nous situons entre ces deux tendances. Du côté ingénieur nous visons à des résultats réels, tout en nous méfiant de l'approche classique qui a causé tant d'échecs dans ce domaine. D'autre part la robotique mobile est probablement la branche de l'ingénieur qui s'approche le plus à la réalité animale. Le robot mobile autonome doit en effet faire face au même monde et au même problème que les animaux. Ainsi, ce travail se concentre sur des modèles proches du biologiquement plausible, en évitant les modèles de traitement de l'information utilisés pour la classification et la reconnaissance de données, par exemple.

Le modèle utilisé principalement dans ce travail est le neurone très simplifié de McCulloch et Pitts [McCulloch43], considéré comme un sommateur d'entrées pondérées (figure 48). On considère ces entrées comme les axones d'autres neurones, les pondérations correspondent aux synapses et le sommateur au corps du neurone avec sa fonction d'intégration. Une fonction de sortie permet de limiter la réponse du neurone dans un domaine spécifique.

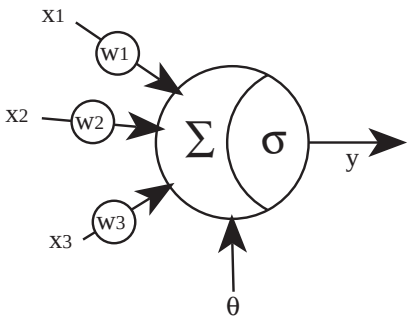


Figure 48: Modèle du neurone de McCulloch et Pitts

On appelle x_i les stimulis d'entrée, w_i les poids synaptiques, θ une valeur de seuil, σ la fonction de sortie et y la valeur de sortie calculée de la façon suivante:

$$y = \sigma\left(\sum_{i=1}^{n} w_i \; x_i \; - \theta\right) \qquad \text{(Eq. 1)}$$

La fonction σ de sortie peut prendre plusieurs formes, mais trois d'entre elles sont les plus utilisées (figure 49).

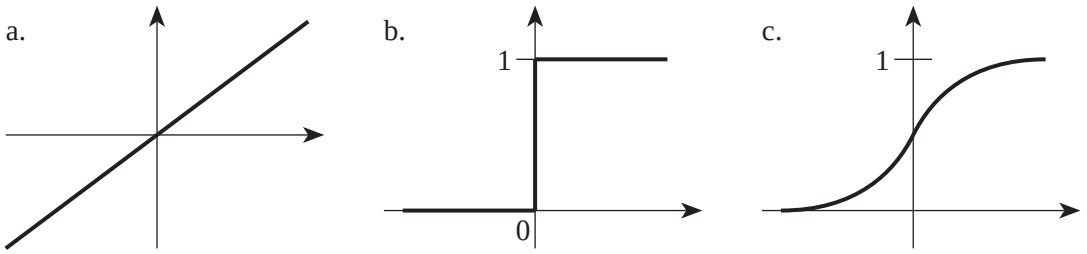


Figure 49: Fonctions de sortie utilisées fréquemment

a. La fonction linéaire

$$\sigma(A) = k\,A \qquad \text{(Eq. 2)}$$

Dans laquelle k est une constante.

b. La fonction échelon:

$$\sigma(A) = \begin{cases} 1 (\text{si } A > 0) \\ 0 (\text{si } A \leq 0) \end{cases} \qquad \text{(Eq. 3)}$$

c. La fonction sigmoïdale:

$$\sigma(A) = \frac{1}{1 + e^{\,-kA}} \qquad \text{(Eq. 4)}$$

Dans laquelle k contrôle l'inclinaison de la courbe: pour $k \rightarrow \infty$ la fonction sigmoïde est égale à la fonction échelon. Cette fonction correspond à la tangente hyperbolique.

Comme dans le monde biologique, les RNA sont également organisés en réseaux. Etant donné N neurones, il est possible de les connecter de différentes façons. Les architectures se distinguent par le nombre de couches de connexions synaptiques à travers lesquelles le signal en entrée doit passer pour rejoindre les neurones de sortie, et par la présence ou non de connexions récurrentes (figure 50).

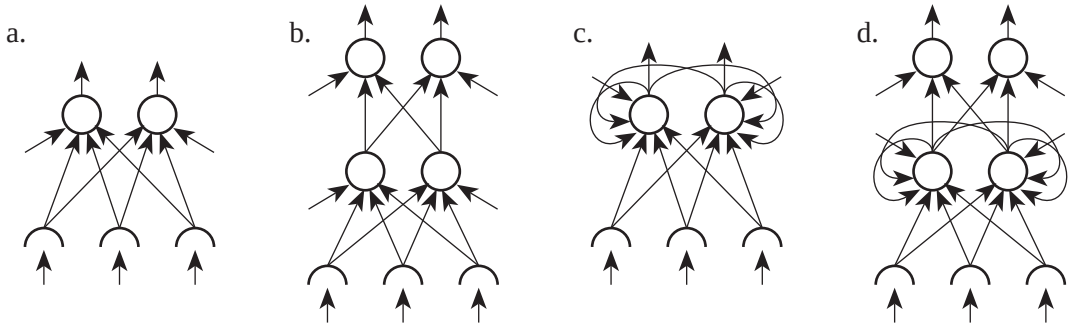


Figure 50: Quelques exemples de structure de réseaux: a) à une couche, b) à deux couches (une couche cachée), c) à une couche avec connexions récurrentes sur la couche de sortie, d) à deux couches avec connexions récurrentes sur la couche cachée. Les entrées latérales sur les neurones représentent les valeurs de seuil

Dans le premier cas de la figure 50, les informations sont introduites par les entrées du réseau (en bas). En robotique mobile, ces informations proviennent généralement du système sensoriel et sont transmises directement aux neurones de sortie, qui, généralement, commandent directement des actionneurs du robot. Le traitement des informations se fait par les connexions pondérées et par la fonction de sortie des neurones de la couche de sortie. Dans le cas d'un réseau à plusieurs couches, un traitement ultérieur est fait par les neurones de la couche intermédiaire, appelée la *couche cachée*, du fait que son activation n'est pas visible de l'extérieur. Des connexions récurrentes peuvent être ajoutées sur n'importe quel neurone de n'importe quelle couche. Deux exemples sont illustrés dans la figure 50 (c. et d.). Dans le cas de réseaux de neurones à temps discret (utilisé dans ce travail), ces connexions donnent, en entrée du neurone au temps (t), l'état des neurones de sa propre couche au temps (t-1). Alors que les sorties des réseaux a) et b) de la figure 50 dépendent uniquement de la valeur d'entrée, les réseaux c) et d) peuvent tenir compte de l'état précédent du réseau. Ces connexions permettent donc de réaliser une sorte de mémoire interne au réseau.

4.2.1 Un exemple de réseau neuronal appliqué à la robotique mobile: les véhicules de Braitenberg

Un exemple très simple d'utilisation de ce type de structure pour le contrôle d'un robot mobile a déjà été donné dans la section 3.2.1, pour réaliser des comportements d'attirance et évitement d'obstacles. La structure de base a été proposée par Valentino Braitenberg dans [Braitenberg84]. Dans son oeuvre, Braitenberg décrit 14 types de véhicules contrôlés par des structures de contrôle imaginaires, ayant toutefois une inspiration biologique neuronale. Une fois présenté le mécanisme de contrôle, Braitenberg imagine le comportement des véhicules et l'analyse de ce comportement faite par une personne ne connaissant pas le mécanisme de contrôle. Par ce type d'expériences imaginaires, il essaie de montrer comment un comportement, qui peut paraître complexe, peut être généré par un mécanisme très simple. Les véhicules 2 et 3 sont parmi les plus simples et sont réalisables techniquement, ce qui n'est pas le cas des autres. Ils exploitent des fonctionnalités sensorielles et motrices similaires à celles disponibles dans Khepera.

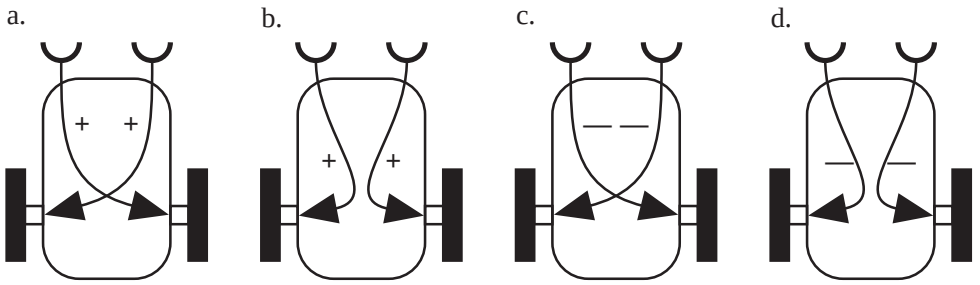


Figure 51: Exemples de véhicules de Braitenberg, dotés de deux roues et de capteurs de lumière sur l'avant

Quelques exemples de ces véhicules sont représentés dans la figure 51. Tous ces véhicules disposent de deux roues à l'arrière et de deux capteurs de lumière à l'avant. Le premier véhicule (figure 51 a) est contrôlé par des connexions excitatrices croisées qui relient les capteurs aux moteurs. Ces connexions font que le véhicule a la tendance à se tourner vers la lumière en accélérant la roue se trouvant du côté opposé à la source de lumière. Sa vitesse est augmentée au fur et à mesure qu'il s'approche de la lumière et que l'intensité de celle-ci s'accroît. Le véhicule b) de la figure 51 évite les sources de lumière. Le véhicule c) a aussi un comportement d'évitement de la lumière, mais en inhibant les roues au lieu de les accélérer. Le véhicule d), enfin, est attiré par les sources lumineuses mais va ralentir en s'approchant de la source, jusqu'à s'arrêter.

Le principe de base de ces véhicules peut être généralisé à N capteurs et à une interconnectivité totale entre capteurs et moteurs comme le montre la figure 52.

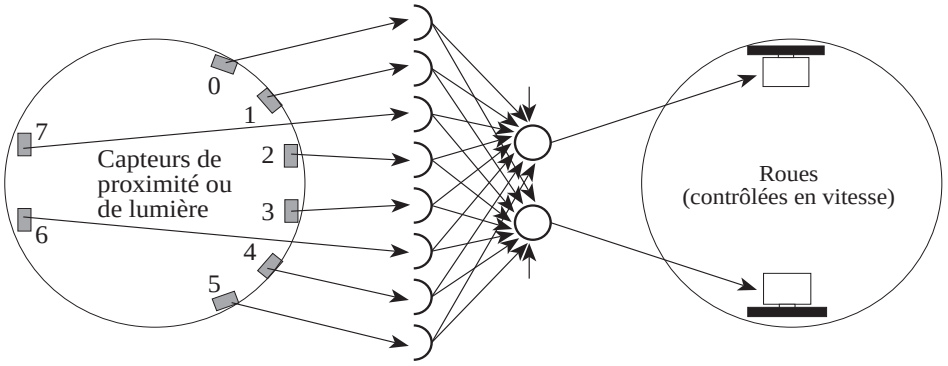


Figure 52: Généralisation du véhicule 3 pour la structure de Khepera

Ce nouveau véhicule généralisé est contrôlé par un réseau de neurones à une couche ayant huit entrées, auxquelles on transmet les valeurs des capteurs de proximité ou de lumière, et deux sorties, que l'on utilise pour commander les vitesses des deux roues du robot. Selon le type des capteurs utilisés (de proximité ou de lumière), le signe et la valeur des pondérations des connexions et de l'entrée de seuil, on peut générer une grande diversité de comportements, allant de l'évitement d'obstacles au suivi de murs, à la recherche d'objets, au suivi de lumière, etc. Deux de ces comportements ont été utilisés pour le comportement collectif décrit dans le chapitre 3, et le type de connexions utilisées a été illustré dans la figure 24.

4.3 Les techniques d'apprentissage

Une des bases de l'apprentissage et de la mémorisation semble être située au niveau du changement des caractéristiques des synapses en fonction de l'activité pré- et post-synaptique. Cette théorie avait déjà été développée par Hebb [Hebb49] et a été confirmée par des travaux récents [Conquet94].

Dans le domaine des réseaux de neurones artificiels, il existe plusieurs algorithmes de modification des poids synaptiques, généralement associés à des types spécifiques de réseaux, à savoir la méthode de rétropropagation du gradient sur des réseaux multicouches [Rumelhart86], le modèle de Hérault et Jutten [Jutten88], les règles de Hebb, le réseau et les méthodes d'auto-organisation de Kohonen [Kohonen88] pour en citer quelques unes. Parmi ces méthodes, certaines sont purement mathématiques et ne se soucient pas de la plausibilité biologique, comme la méthode de rétropropagation du gradient (voir [Crick89]), par exemple. Le but de ce type d'algorithmes est de trouver les poids synaptiques du réseau afin d'obtenir une fonction bien définie. Or, ces algorithmes, en bonne partie à cause de leur manque de plausibilité biologique, posent beaucoup de problèmes lors de leur utilisation sur un système réel:

- Plusieurs de ces méthodes, comme la rétropropagation du gradient, impliquent qu'un maître connaît la réponse, soit partiellement, soit complètement, ce qui est difficile à réaliser pour des systèmes complexes et autonomes se déplaçant dans le monde réel.
- Beaucoup de méthodes (rétropropagation du gradient, réseau de Kohonen, par exemple) différencient la phase d'apprentissage de la phase d'exploitation du réseau. Or, cette différentiation ne peut pas exister si on veut obtenir un système continuellement adaptatif.
- Pour certaines méthodes, lors de l'apprentissage, l'ordre et la distribution des exemples joue un rôle clé (réseau de Kohonen). Ceci crée un vrai problème lors d'applications réelles.
- Le nombre d'itérations pour atteindre la convergence est souvent tellement élevé qu'il rend impossible toute implémentation réelle.

Ces méthodes, qui s'appliquent très bien à des problèmes d'optimisation ou de modélisation de fonctions non-linéaires, trouvent donc difficilement une application dans la robotique mobile réelle. Pour cette raison, dans ce travail, l'attention a été portée plutôt vers les méthodes plus biologiquement plausibles et les plus facilement applicables à des systèmes adaptatifs, à savoir les règles de Hebb.

Toute règle de modification des poids synaptiques est basée sur l'équation

$$w_{ij}^{t} = w_{ij}^{t-1} + \Delta w_{ij}^{t} \qquad \text{(Eq. 5)}$$

Dans laquelle w_{ij}^{t} est le poids de la connexion entre le neurone présynaptique i et le neurone post-synaptique j au temps t, w_{ij}^{t-1} est le même poids au temps (t-1), Δw_{ij}^{t} est la modification des poids au temps t. La valeur des poids n'est pas bornée.

La modification des poids dépend du type de règle appliquée. La règle de Hebb de

base renforce les poids d'une connexion entre deux neurones actifs. Elle se formule de la façon suivante:

$$\Delta w_{ij} = x_i \; y_j \qquad \text{(Eq. 6)}$$

Avec x_i représentant l'activation du neurone pré-synaptique i, y_j l'activation du neurone post-synaptique j. La valeur des activations des neurones est normalisée entre 0 et 1. La grande limitation de cette règle est qu'elle ne prévoit pas de réduction des poids synaptiques. Ceci peut facilement mener les poids à une forte croissance, qui provoque une plus forte activité dans le réseau, qui, à son tour, provoque une croissance des poids encore plus forte et ainsi de suite. Ce problème est partiellement résolu par la règle post-synaptique, qui ajoute à la règle de Hebb de base une diminution du poids lorsque le neurone post-synaptique est actif alors que le neurone présynaptique ne l'est pas. Elle s'exprime par la relation suivante:

$$\Delta w_{ij} = x_i \; y_j + (y_j - 1)x_i \qquad \text{(Eq. 7)}$$

La règle pré-synaptique prévoit un mécanisme similaire, mais symétrique, réduisant la valeur du poids lorsque le neurone pré-synaptique est actif alors que le neurone post-synaptique ne l'est pas:

$$\Delta w_{ij} = x_i \; y_j + (x_i - 1)y_j \qquad \text{(Eq. 8)}$$

La règle de Hebb généralisée, ou la règle de Hopfield [Hopfield82], enfin, combine et généralise les deux règles précédentes: elle renforce le poids, quand les neurones pré- et post-synaptiques sont dans le même état et le réduit, quand les deux neurones sont dans des états différents.

4.4 Un exemple d'application en robotique mobile

Une structure qui utilise la règle hebbienne pour réaliser une adaptation continue à l'environnement est le *Distributed Adaptive Control* (DAC). Ce modèle a été présentée et étudiée en simulation par P. Verschure [Verschure92]. Il s'agit d'un modèle distribué et auto-organisé inspiré du phénomène biologique du conditionnement classique [Verschure91]. Ce modèle se base sur une structure neuronale qui contient implicitement les principes fondamentaux de comportement du robot. Ces principes sont codés par l'expérimentateur et dirigent le processus d'apprentissage.

La structure neuronale est établie de façon à prendre en compte les caractéristiques des capteurs et des actionneurs, comme de la morphologie de l'agent. Les principes de comportement de l'agent sont codées dans cette structure neuronale sous forme de réflexes de base, comme par exemple le fait de reculer et tourner à droite lors d'une collision avec un obstacle sur la gauche. Ces réflexes sont réalisés dans le réseau de neurones par des connexions reliant directement les capteurs avec les unités motrices, et déterminent entièrement les actions du robot dans ses premières phases de vie. Pour permettre au

système de s'adapter à l'environnement, le système de contrôle dispose de données provenant de capteurs distaux, qui donnent à l'agent plus d'informations sur l'environnement. L'adaptation du système à l'environnement se fait à travers une intégration progressive et continuellement revue de ces capteurs dans le comportement du robot. Cette prise en compte est basée sur l'interaction robot-environnement et est dirigée par les réflexes de base. Le mécanisme d'intégration est la règle hebbienne corrigée par un terme d'oubli. Dans le cadre de ce travail, cette théorie a été transposée sur un robot réel afin de mettre en évidence ses caractéristiques dans des problèmes d'évitement d'obstacles et de vision simplifiée. La section suivante permet de clarifier les détails de fonctionnement de ce modèle et illustre son fonctionnement sur un robot réel.

4.4.1 La structure du réseau

Pour mettre en évidence les possibilités et les limitations de cette approche, nous avons choisi de mettre en oeuvre un expérience basée sur le comportement d'évitement d'obstacles. Ce comportement se prête très bien aux comparaisons, car il a été obtenu avec plusieurs techniques.

La structure neuronale DAC utilisée dans notre expérience est représentée dans la figure 53. Elle comprend essentiellement deux branches:

1 La branche *réflexe*: Elle relie les capteurs de collision (en haut à gauche) aux neurones moteurs par les neurones du groupe US (expliqué au point 2). Les capteurs de collision ont une sortie binaire. Les connexions qui forment cette branche sont fixes, à coefficient égal à un, et déterminées par le concepteur. Elles traduisent les principes de comportement du robot: si le robot touche un objet sur la droite, il tourne vers la gauche et vice-versa.

   Les neurones qui forment cette branche sont des neurones à sortie binaire, ayant donc à leur sortie une fonction échelon caractérisée par un seuil. L'activation d'un capteur de collision provoque l'activation d'un neurone de la couche US (pour *Unconditioned Stimulus*), qui est considérée comme une réponse inconditionnelle. Cette activation provoque le déclenchement d'une action motrice. Les neurones moteurs sont trois et déclenchent trois actions distinctes: éviter à droite, éviter à gauche et avancer. L'action d'évitement est composée par un recul et une rotation sur place dans la direction indiquée. Les activations provenant de la couche US activent uniquement les deux neurones responsables de l'évitement. Lorsqu'aucun de ces deux neurones est actif, le neurone responsable de l'avancement est activé par défaut. Lorsqu'un des deux neurones est activé, il prend la main. Lorsque les deux neurones d'évitement sont activés, un des deux prend le contrôle, le choix étant fait de façon aléatoire.

2 La branche *de conditionnement*: Elle relie des capteurs d'obstacles de type distal (en bas à gauche sur la figure 53) à la couche des neurones US. Ce deuxième groupe de capteurs détecte des caractéristiques plus fines de

l'environnement qui sont en relation avec la présence d'obstacles et qui sont perçues en avance par rapport aux capteurs de collision. Ces capteurs peuvent être des capteurs de distance (comme expérimenté en simulation par Verschure) ou un système plus sophistiqué comme de la vision (utilisé dans ce travail).

Les neurones présents à l'entrée de cette branche forment la couche CS (pour *Conditioned Stimulus*). Ces neurones reçoivent en entrée les mesures normalisées provenant des capteurs et ont une fonction de sortie linéaire. Cette couche est reliée point à point avec la couche US. Ces connexions, initialisées à zéro, sont soumises à une règle hebbienne modifiée (Eq. 10).

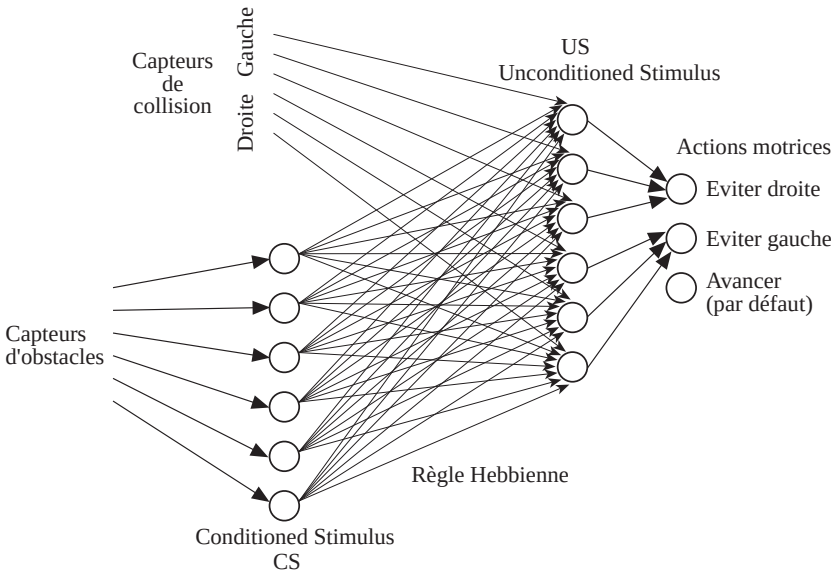


Figure 53: Structure de contrôle du robot basée sur le modèle DAC

L'activation du neurone j de la couche US est définie par la relation suivante:

$$a_j = c_j + \sum_{i=1}^{N} w_{ij} \; x_i \qquad \text{(Eq. 9)}$$

Dans laquelle c_j est l'entrée binaire provenant des capteurs de collision, w_{ij} est le poids de la connexion reliant le neurone i de la couche CS au neurone j de la couche US, x_i est la sortie du neurone i de la couche CS, N est le nombre de neurones de la couche CS. Les poids w_{ij} sont mis à jour selon la règle suivante:

$$\Delta w_{ij} = \frac{1}{N}(\eta \; x_i \; y_j - \varepsilon \; \bar{x} \; w_{ij}) \qquad \text{(Eq. 10)}$$

Dans laquelle

- x_i est la sortie du neurone i du groupe CS

- y_j est la sortie du neurone j du groupe US
- w_{ij} est le poids de la connexion reliant le neurone i de la couche CS au neurone j de la couche US
- $\bar{x}$ est la valeur moyenne de la couche US
- N est le nombre de neurones de la couche CS
- η est le gain d'apprentissage
- ε est le gain d'oubli.

Le premier terme de cette équation correspond à une règle hebbienne pondérée par un gain. La deuxième partie introduit un terme d'oubli, qui permet de réduire la valeur du poids synaptique, si celui-ci n'est plus activé et que d'autres poids le sont (donné par le terme $\bar{x}$).

La structure du réseau de la figure 53 et les équations illustrées ci dessus permettent de réaliser le comportement de conditionnement illustré au début de cette section 4.4: Dans une première phase de la vie du robot, le comportement du robot est totalement défini par les connexions de la voie réflexe. Lorsqu'une collision a lieu et qu'un neurone de la couche US est activé, un renforcement est réalisé sur les connexions qui relient ce neurone à des neurones actifs dans la couche CS. Si cette situation se reproduit régulièrement, c'est-à-dire si des neurones de la couche CS sont représentatifs d'une détection d'obstacle, la connexion correspondante entre couche CS et UR se renforce. Si, par contre, cette situation se présente de façon aléatoire et ne représente donc pas une correspondance entre capteurs distaux et de collision, le renforcement de la connexion est contrebalancé par l'oubli.

L'apprentissage commence à influencer le comportement lorsqu'une connexion entre CS et UR devient suffisamment importante pour que l'activité du groupe CS déclenche à elle seule l'activité d'un neurone de la couche UR, provoquant une action de la part du robot. A ce moment, le robot commence à utiliser ses capteurs distaux pour détecter les obstacles.

4.4.2 Les données de l'expérience

Afin de tester le fonctionnement de ce type de réseau, il est nécessaire de disposer de capteurs de collision ainsi que de capteurs distaux, ces derniers pouvant détecter les obstacles. N'ayant pas de vrais capteurs de collision sur le Khepera, nous avons utilisé les capteurs de distance seuillés à une valeur très élevée. Ceci permet d'obtenir une valeur binaire qui devient active lorsqu'un objet se trouve à une distance du robot de moins de 15 mm.

Pour réaliser les capteurs distaux, nous avons choisi d'utiliser la vision. Afin de pouvoir utiliser la vision pour la détection d'obstacles, il faut des caractéristiques visuelles qui permettent de reconnaître les obstacles, voir d'en estimer la distance. Ceci est possible en utilisant des objets striés, avec une fréquence spatiale de stries constante. Ainsi les murs ont été recouverts de stries verticales noires de 4 mm de largeur et séparées de 4 mm de blanc, comme illustré dans la figure 54. Cette fréquence spatiale a été choisie en fonction des possibilités de détection du système visuel, afin que celui-ci

puisse détecter une variation de fréquence pour des distances du mur allant de 5 mm à 10 cm environ.

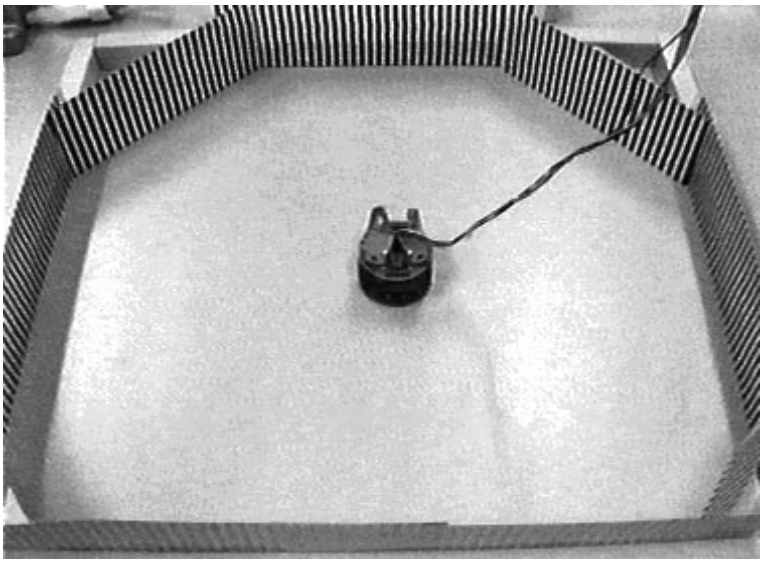


Figure 54: Environnement utilisé pour l'expérience d'apprentissage, avec le robot au milieu. La taille réelle est d'environ 58 x 48 cm.

Au niveau des capteurs visuels, le robot a été équipé de deux caméras linéaires de 64 pixels chacune, disposées comme le montre la figure 55. Chaque caméra a un angle de vue de 35 degrés, et l'axe des deux caméras a été placé à environ 13 degrés de l'axe principal d'avancement du robot. Les images obtenues de ce système sont formées de 64 pixels ayant 256 niveaux de gris chacun. Un capteur de luminosité ambiante, placé entre les deux caméras, ajuste, en permanence, la sensibilité des caméras, ce qui correspond à la fonctionnalité d'iris automatique. Ceci permet d'exploiter au mieux les caractéristiques des capteurs en fonction de la luminosité.

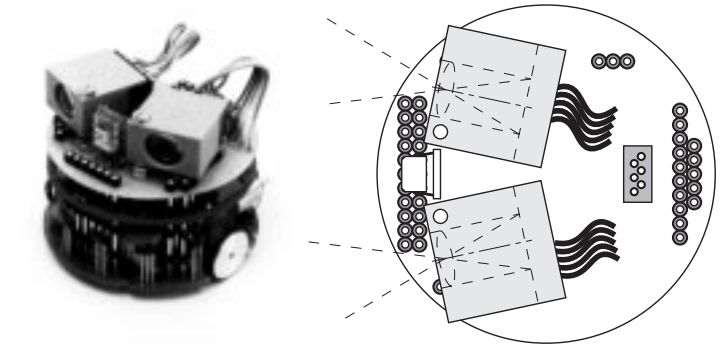


Figure 55: Photo du robot Khepera équipé du système de vision et vue du haut du positionnement des deux caméras linéaires

Dans l'environnement décrit plus haut, les caméras linéaires permettent d'obtenir une section horizontale de l'environnement strié. Le type d'image obtenue de ce système placé face à un obstacle est décrit dans la figure 60. A cause de l'optique très imprécise

de ce module de vision, les rayes projetées sur le capteur linéaire donnent lieu à des images plus proches à de sinusoïdes qu'au signal carré idéal.

Afin de pouvoir mesurer la présence et la distance des obstacles à partir des images des caméras, cette entrée visuelle a été filtrée en utilisant plusieurs filtres spatiaux, sensibles à des fréquences spatiales différentes. Ces filtres spatiaux ont été implémentés en utilisant des réseaux de neurones: Chaque valeur d'intensité d'un pixel a été introduite comme entrée d'un réseau à une couche de connexions fixes reliant 32 entrées à 32 neurones de sortie, comme le montre la figure 56. Au lieu de considérer tous les 64 pixels disponibles sur la caméra, seulement un pixel sur deux (32 au total) a été pris en compte pour cette expérience. Ce choix a été fait principalement à cause du temps d'exécution des calculs de filtrage, ce calcul étant onéreux car exécuté sur le processeur du Khepera même.

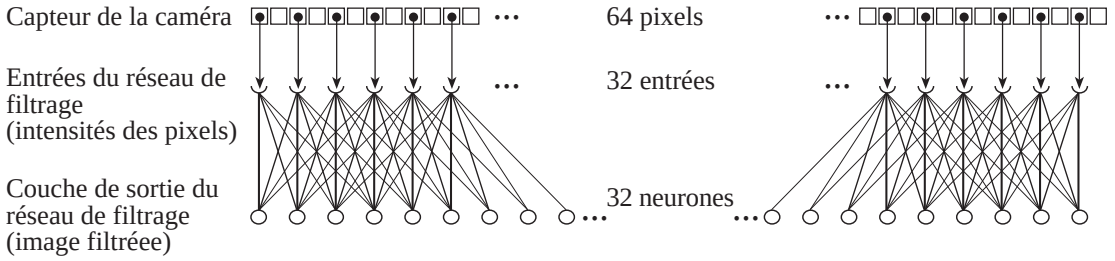


Figure 56: Structure du réseau de neurones de filtrage et connexion au capteur linéaire de la caméra.

Afin de réaliser un filtre, les poids des connexions doivent dépendre de la distance spatiale du neurone de sortie par rapport a celui d'entrée, comme le montre la figure 57, et cette dépendance doit suivre une loi en forme de *chapeau mexicain*:

$$w(d) = ae^{-a|d|} - be^{-b|d|} \qquad \text{(Eq. 11)}$$

Dans laquelle d représente la distance entre le neurone de sortie et celui d'entrée et les paramètres a et b permettent de définir la fréquence de filtrage.

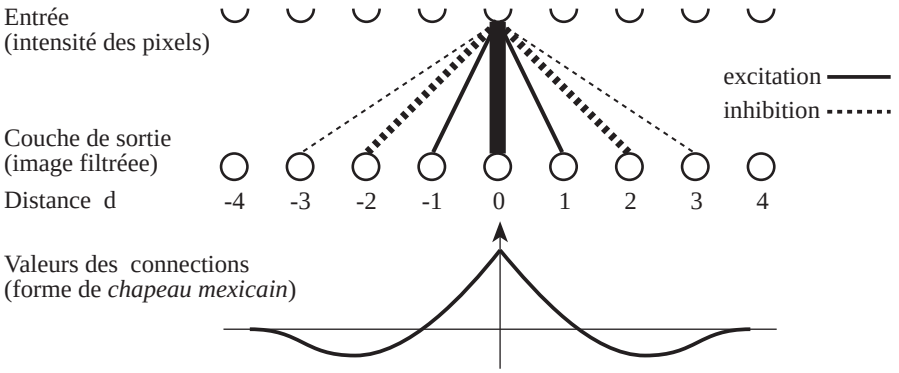


Figure 57: Connexions de filtrage entre les entrées correspondantes aux intensités des pixels et la couche des neurones de sorties, correspondant à l'image filtrée. Les poids des connexions suivent une courbe en *chapeau mexicain*

Avec de telles interconnexions, cette couche de traitement neuronal effectue un filtrage spatial passe-bande (pour une analyse approfondie voir [Hérault92], page 40).

Quatre de ces couches ont été implémentées afin de détecter de façon grossière quatre gammes de fréquences appelées *hautes fréquence* (hf), *moyennes hautes fréquences* (mhf), *moyennes basses fréquences* (mbf) et *basses fréquences* (bf). Ces quatre couches sont toutes reliées à la même entrée, représentant les intensités des pixels. Leur sorties représentent les parties des images qui contiennent la fréquence spatiale appartenant à la bande sur laquelle le filtre est spécialisé.

La figure 58 représente les valeurs des connections de ces quatre couches. Ces valeurs sont représentées en fonction de la distance du neurone de sortie par rapport au neurone d'entrée, selon la notation représentée dans la figure 57. Les valeurs numériques ont été choisies de façon volontairement approximative, tout en mimant une fonction en chapeau mexicain avec ses caractéristiques de base (intégrale nulle, points de croisement avec l'axe des x espacés selon la fréquence).

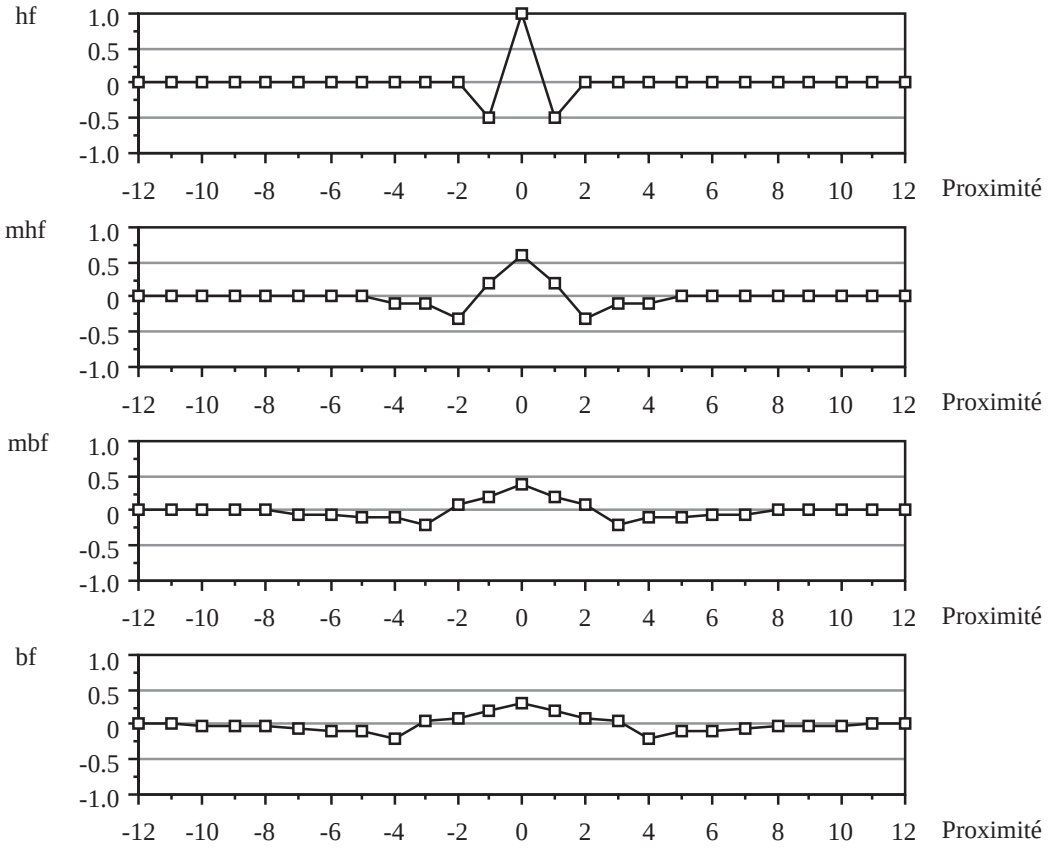


Figure 58: Valeurs des connexions entre un neurone de la couche d'entrée visuelle et la couche de sortie filtrée, en fonction de la proximité du neurone de la couche de sortie et pour les quatre couches de traitement (hf, mhf, mbf et bf)

Une méthode simple pour arriver à déterminer la présence ou pas d'une certaine fréquence spatiale dans une image en ayant à disposition la sortie des filtres, est de quantifier l'activité de chaque couche de sortie correspondant à un filtre. Ceci a été réalisé en

calculant la moyenne des activités de tous les neurones de chaque couche. Le calcul de la moyenne correspond à une couche neuronale supplémentaire ayant une pondération uniforme et terminant sur un seul neurone de sortie. Comme le montre la figure 59 on obtient ainsi quatre neurones (appelés *hf*, *mhf*, *mbf* et *bf* en correspondance des filtres auxquels ils sont associés) qui quantifient la présence des différentes fréquences spatiales dans l'image perçue par la caméra. A ces neurones on a ajouté une fonction de sortie qui permet de mieux exploiter la dynamique de l'activité obtenue par le moyennage. En effet le maximum de cette activité ne dépasse jamais la valeur 0.5 et atteint rarement des valeurs supérieures à 0.3. D'autre part une activité est toujours présente, ce qui donne un minimum d'activité se situant autour des valeurs 0.5 à 0.1. Ainsi la fonction de sortie normalise entre 0 et 1 le domaine d'activité allant de 0.1 à 0.3.

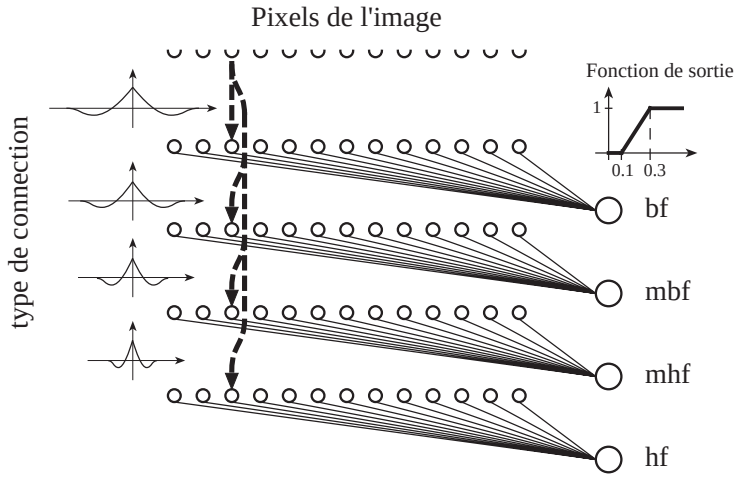


Figure 59: Structure de filtrage basée sur quatre filtres passe-bande et quatre neurones de sortie. Ces neurones moyennent et normalisent l'activité des couches de sortie des filtres respectifs

L'activation de ces quatre neurones de sortie représente donc une sorte de spectre de l'image présente sur la caméra. La figure 60 illustre quatre exemples d'images et d'activités des quatre neurones associées.

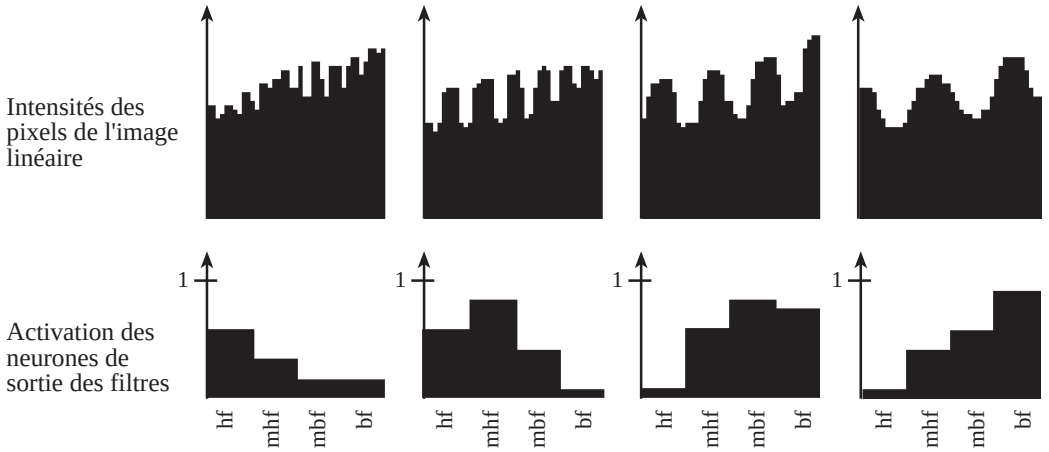


Figure 60: Quatre exemples d'images provenant de la caméra, avec les niveaux de sortie des neurones de détection de fréquences spatiales

On peut observer que les réponses donnent une indication de la fréquence prédominante dans l'image. Cette prédominance n'exclut toutefois pas la présence d'autres fréquences, les activités des autres neurones n'étant jamais nulles.

La structure de la figure 59 a été répétée deux fois, une fois pour chaque caméra du robot. Nous avons donc ainsi obtenu 8 neurones qui forment le groupe de neurones CS de la structure neuronale DAC présentée dans la figure 53. Le but de l'apprentissage est donc d'associer l'activité de cette couches (liée à la présence d'obstacles grâce leur caractéristiques visuelles) à celle de la couche d'évitement d'obstacles afin de réaliser ce comportement à l'aide du système visuel uniquement.

4.4.3 Résultats

Un large nombre d'essais a été fait afin de trouver une bonne combinaison des trois paramètres principaux de l'algorithme, à savoir le seuil des neurones du groupe US, le gain d'apprentissage η et le gain d'oubli ε. Le choix de ces deux derniers paramètres est particulièrement important, car leur interaction détermine la stabilité des poids du réseau. D'autre part le choix du gain d'apprentissage et du seuil des neurones du groupe US détermine la vitesse d'apprentissage.

Les expériences ont montré qu'un bon choix de paramètres pour la stabilité des poids est une valeur des seuils à 0.2, $\eta = 0.1$ et $\varepsilon = 0.9$. Cette stabilité n'est toutefois que relative et limitée dans le temps. Des expériences de longue durée ont toujours mené à une croissance générale des poids du réseaux qui provoque une généralisation de l'évitement à n'importe quelle fréquence spatiale. Cette combinaison de paramètres permet de limiter cet effet, mais donne lieu à des apprentissages très lents, nécessitant environ 20-30 collisions avant de pouvoir commencer à réaliser des détections visuelles.

En augmentant le gain d'apprentissage à une valeur de 1.0 on peut obtenir un apprentissage rapide qui aboutit au comportement d'évitement d'obstacles visuel après quelques interactions avec l'environnement. La figure 61 illustre un cas d'apprentissage rapide utilisant ces paramètres. Le robot commence par effectuer deux collisions frontales avec les murs. Pendant ces premières expériences les connections entre la couche CS et la couche US sont renforcées jusqu'à causer le déclenchement des neurones moteurs par le seul stimuli visuel. C'est le cas, par exemple, des trois évitements successifs. Lorsque le robot recommence à avancer et rencontre le mur à nouveau, le mur apparaît sous un autre angle de vue, donnant lieu, sur la rétine du système visuel, à une autre fréquence spatiale. Cette dernière n'est pas reconnue et il y a collision entre robot et mur. Ce type de collision latérale-avant a lieu deux fois avant que le robot ne fasse une détection visuelle, juste après ces deux collisions et se situant en bas à droite dans la figure 61. Dans la suite du trajet le robot a encore des collisions lui permettant de consolider les poids appris et d'apprendre les détections purement latérales. C'est seulement lors de son passage sur la gauche de l'environnement que les détections se font uniquement visuellement.

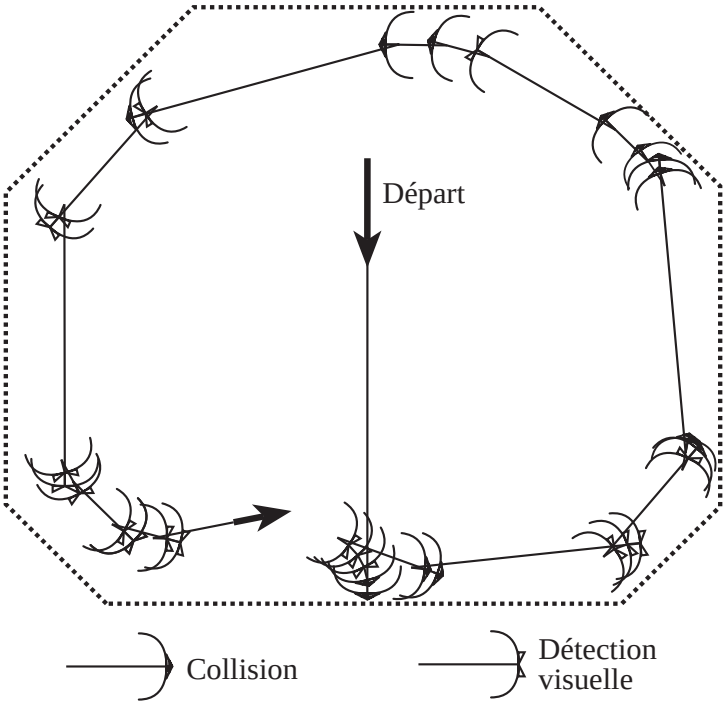


Figure 61: Trajectoire du robot et réactions face aux murs

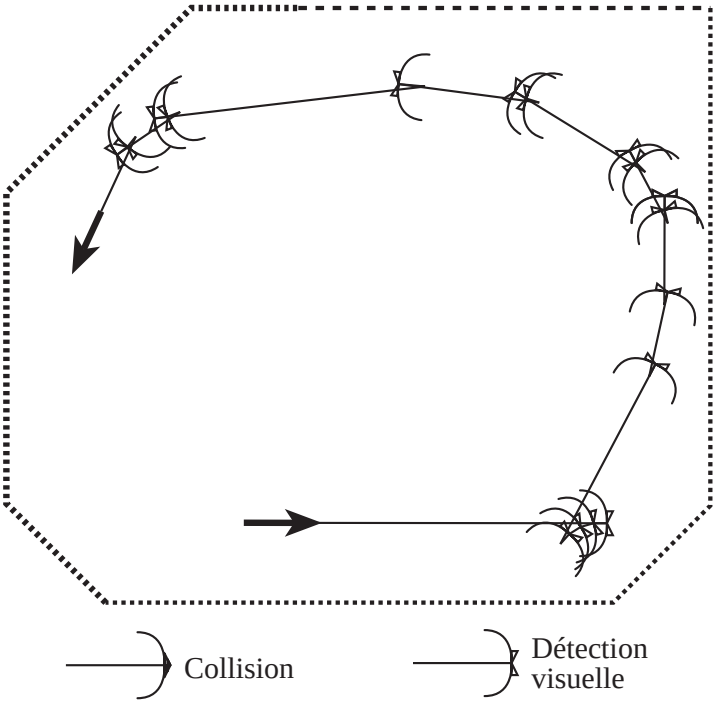


Figure 62: Réaction du robot face à un changement d'environnement: Le mur en haut à droite a été strié avec une fréquence spatiale plus basse, ce qui trompe le système visuel et cause des évitements à une plus grande distance

La réponse aux stimuli visuels a été testée, après une première phase d'apprentissage de deux minutes et 30 secondes (environ 60 itérations d'apprentissage), dans l'environnement présenté dans la figure 62. La description du comportement donnée dans cette figure a été mesuré lors de la même expérience que la figure précédente, mais après que le robot ait parcouru un tour supplémentaire de l'environnement.

L'environnement de la figure 62 est identique à celui présenté dans la figure 61, à différence du mur en haut à droite, dont les bandes verticales ont une fréquence spatiale plus basse (la moitié) que le reste de l'environnement. Lorsque le robot se trouve en face de ce mur, il l'évite à une distance bien supérieure à celle habituelle. Ce mur à cette distance apparaît, en effet, optiquement similaire à un autre mur mais vu de plus près. Ceci prouve bien le mécanisme de détection de distance basé sur les fréquences spatiales.

Le grand désavantage de l'utilisation d'un gain d'apprentissage élevé réside dans la généralisation rapide de l'évitement à n'importe quelle fréquence spatiale. Les poids du réseau tendent en effet à croître fortement au niveau de tout le réseau. Ce phénomène, associé à celui de l'auto-excitation due à la règle de Hebb, donne lieu à des reconnaissances visuelles de plus en plus généralisées, qui portent à un système qui reconnaît n'importe quelle fréquence spatiale comme étant un obstacle.

Lors de l'utilisation de paramètres plus stables (valeur des seuils à 0.2, $\eta = 0.1$ et $\varepsilon = 0.9$) on peut mieux observer la croissance des poids et leur signification. La figure 63 illustre un exemple de la matrice d'interconnexion entre les neurones du groupe CS et la moitié des neurones du groupe US responsables de la rotation à gauche. Cette matrice a été obtenue lors d'une expérience similaire à celle illustrée dans la figure 61.

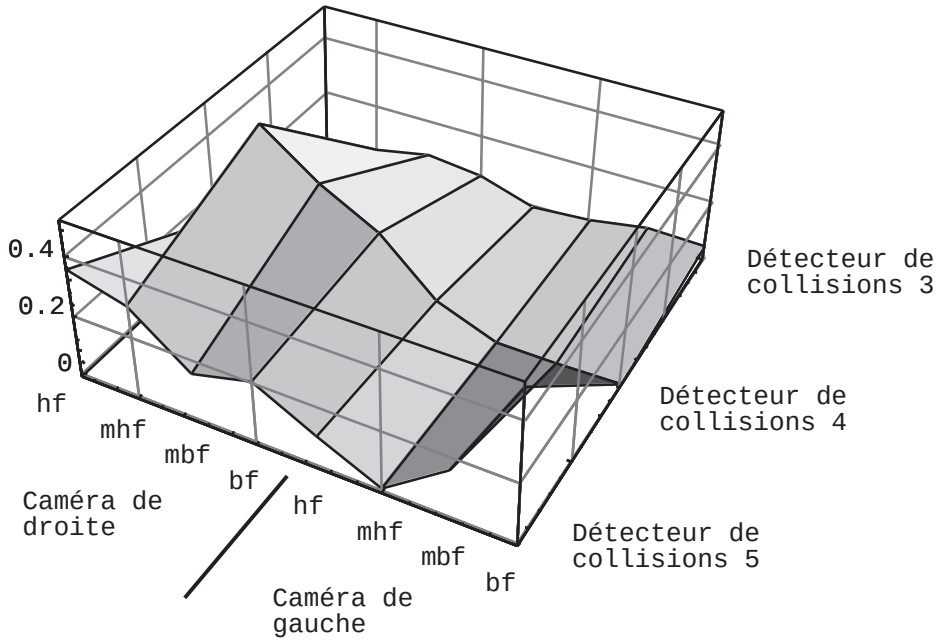


Figure 63: Représentation des interconnections entre le groupe CS et la moitié du groupe US (capteurs de collision à droite, numérotés selon la figure 52) créées par l'apprentissage effectué dans l'environnement présenté en figure 54

On peut observer que l'algorithme a trouvé une corrélation entre le fait de détecter un objet avec le capteur numéro 4 (selon la figure 52) et le fait d'avoir une fréquence moyenne-haute sur sa caméra de droite. Les fréquences plus basses sont aussi présentes, mais en proportion mineure. Lors de collisions avec des obstacles pris sous un angle plus élevé (collision détectée par le capteur numéro 5), l'association est faite avec des fréquences plus élevées, ce qui correspond bien à la vue des obstacles que le robot a dans une telle situation.

Une autre association assez forte est faite entre les collisions avec des obstacles se trouvant sur le flanc droit (capteur 5) et une présence de basses fréquences sur la gauche. Ceci est principalement dû a un manque d'image nette sur la caméra de gauche lors de telles collisions.

4.4.4 Discussion

La méthode de conditionnement utilisée dans cette expérience a la particularité, par rapport à d'autres méthodes d'apprentissage, de permettre de construire un comportement complexe, ou basé sur des capteurs complexes, à partir de principes simples codés dans les réactions de base du robot (évitement d'obstacles lors de collisions). Ces principes codés dans les réactions de base ont une fonction similaire, par exemple, aux règles de renforcement utilisé dans [Touzet96]. L'aspect très important, et qui est commun à la plupart des approches d'apprentissage, est que ce comportement est construit sur la base de l'interaction robot-environnement. La structure d'apprentissage permet en effet de saisir les caractéristiques de l'environnement qui lui sont utiles en modifiant les poids qui règlent le comportement du robot. La présence d'un terme d'oubli, plus particulière à cette règle d'apprentissage, permet une remise en question continue des connaissances afin de les garder à jour.

Au delà du fonctionnement correct mis en évidence dans la section précédente, ce type de structure se base sur un équilibre très délicat entre apprentissage et oubli, et présente le problème d'auto-excitation dû à la règle hebbienne. Ces problèmes de base sont amplifiés par le fait que les différents filtres ont des chevauchements sur les fréquences, ce qui cause des activités simultanées non nulles sur les différents neurones de détection de fréquence (voir figure 60). L'auto-excitation peut ainsi se propager d'un neurone à l'autre à travers tout le réseau.

Ce problème, qui apparaît de façon claire dans cette expérience, n'était pas tellement gênant pour les expériences de Paul Verschure. Lors de ses simulations il utilisait, comme entrées du groupe de neurones CS, la valeur de capteurs de distance ayant une caractéristique très similaire à celle des capteurs de proximité de Khepera (voir la figure 15 à la page 40 et [Verschure92b]). Or, les poids synaptiques de la structure DAC, dans ce cas, ne font rien d'autre qu'introduire un seuil entre l'activité de ces capteurs et celle des neurones du groupe US. L'apprentissage se réduit donc à l'augmentation de ce seuil, ce qui est très peu sensible à la vitesse d'apprentissage et ne souffre pas de l'effet d'auto-excitation. Au contraire, une auto-excitation appliquée à ce type de structure ne fait qu'améliorer les performances du système.

L'utilisation d'entrées de ce type (valeurs continues correspondant à une grandeur

physique de l'environnement) simplifie la tâche et l'utilisation de la structure neuronale, mais n'est pas du tout généralisable. Il suffit en effet d'avoir d'autres capteurs de distance (qui, par exemple, rendent une valeur proportionnelle à la distance), et la structure d'apprentissage n'est plus en mesure de faire des associations. Dans les systèmes biologiques, il n'existe d'ailleurs que peu de structures neuronales dans lesquelles des neurones sensoriels sont actifs proportionnellement à une grandeur physique. Dans la plupart des cas les neurones s'activent en correspondance à la présence d'un certain stimulus bien caractérisé: Dans la cochlée, par exemple, il n'existe pas de neurone qui s'active proportionnellement à la fréquence du son entendu, mais la structure est composée par un ensemble de neurones, chacun actif à une certaine fréquence. C'est ce type de structure qui a été implémenté dans le système visuel décrit plus haut. Avec ce genre d'entrée, le réseau DAC est en mesure de faire des associations bien plus générales entre les grandeurs physiques mesurées et les actions à entreprendre. Toutefois, dans ces conditions, le problème d'auto-excitation des connexions devient très important.

Lors de ses derniers travaux sur le DAC, Paul verschure a proposé des améliorations de la structure [Verschure92b] afin de limiter les effets d'auto-excitation. D'autres extensions de la structure DAC ont étés proposées afin d'atteindre des comportements plus complexes, mais toujours basés sur le conditionnement.

Malgré les améliorations que l'on peut apporter à ce modèle, il reste de gros problèmes liés au choix des capteurs, des paramètres, des règles et de la structure du réseau. Si on fait un mauvais choix de capteur (un capteur qui est normalement actif) et que celui-ci excite continuellement un neurone, cette activation causera un apprentissage qui n'a pas de raison d'être et qui causera des comportements pathologiques. Il est donc nécessaire de choisir soigneusement ses capteurs et l'utilisation que l'on fait des données que l'on en tire. Les paramètres de l'apprentissage sont un autre exemple de choix délicat: L'équilibre entre le gain d'apprentissage et celui d'oubli est très difficile à atteindre, mais est essentiel si l'on veut obtenir une mise à jour continue des connaissances du réseau. Verschure et Almassy ont essayé de surmonter ce problème par l'utilisation d'algorithmes génétiques [Almassy92]. Enfin la structure du réseau (figure 53) est très peu flexible et conditionne fortement le type de comportement que l'on peut s'attendre, limitant fortement l'autonomie du robot.

4.5 Discussion du problème de l'autonomie

Si cet algorithme possède une certaine autonomie dans son choix des caractéristiques pertinentes observées dans l'environnement, la même autonomie est totalement absente à tous les autres niveaux. L'émergence du comportement appris a lieu dans une structure très rigide, dans laquelle la partie de conception du développeur joue un rôle prépondérant. Cette structure va jusqu'à influencer fortement l'apprentissage, au point que celui-ci se réduit à une simple adaptation de paramètres d'un mécanisme strictement prédéfini. L'apprentissage se fait en effet sur des capteurs bien calibrés pour qu'ils aient un sens, bien pré-traités pour qu'ils puissent être utilisés par la règle d'apprentissage, dans le cadre d'une structure bien ajustée, etc. Le système visuel utilisé dans cette expérience est un bon exemple: l'environnement a été modifié pour permettre une détection

par le système visuel, ce dernier a été ajusté pour saisir les bons paramètres et les présenter sous la bonne forme au réseau DAC proprement dit. L'apprentissage se limite finalement à ajuster une très petite partie de l'ensemble des paramètres du système. Son autonomie est ainsi très limitée, même s'il s'agit d'un premier pas dans la bonne direction.

Le manque d'autonomie n'est pas un problème en soi sans retombées pratiques. Comme mentionné plus haut, l'importance que le concepteur a dans le choix de la structure, des capteurs, etc. fait que les caractéristiques apprises par le système sont celles que le concepteur voit. Il est presque impossible que le système développe des capacités n'ayant pas été envisagées par le concepteur, simplement parce que le système ne dispose pas d'autres moyens que ceux soigneusement mis en place par le concepteur. Ceci cause le problème bien connu en intelligence artificielle classique du *frame of reference*. Le concepteur projette dans le système de contrôle sa façon de voir le monde, sa façon d'analyser les données et ses connaissances des caractéristiques de l'environnement. Or, la complexité de l'environnement est souvent hors de la portée du concepteur et ses méthodes d'analyse trop restreintes: Nos façons de traiter et comprendre les données sont dans la plupart des cas limitées aux systèmes linéaires, alors que, dans le monde réel, dans les capteurs et les actionneurs, il n'y a que très peu de linéarités. Ceci fait que les structures mises en place sont très souvent bien plus complexes que celles dont le robot a besoin, et mal adaptées à la tâche et aux fonctionnalités du robot. Ainsi il est essentiel qu'une partie de ces structures soit créée en fonction de l'interaction entre le robot et son environnement.

Le fait de donner au système de contrôle une direction de développement, ici sous forme de réflexes de base, est nécessaire. Comme nous l'avons vu au chapitre précédent, il est nécessaire de diriger le développement, et celle-ci est une des façons possibles. Le problème de cette méthode est que la spécification de la direction est très précise au point de définir quels neurones doivent être activés par quels autres. Celle-ci est une autre forte limitation de l'autonomie du système de contrôle, car elle ne laisse pas une grande marge de manoeuvre au développement du système de contrôle. L'autonomie ne consiste pas seulement dans la prise de décision, mais aussi et surtout dans la création des structures qui permettent la prise de décision.

C'est à ce niveau que le plus grand travail doit être fait. La méthode de conception doit laisser au système une grande marge de manoeuvre, tout en dirigeant le développement dans la direction souhaitée par le concepteur. Le chapitre suivant se penche sur la méthode évolutionniste qui se propose d'aller dans cette direction.

Enfin, aussi cette expérience prouve que pour pouvoir mettre en évidence les problèmes ainsi qu'en étudier les solutions possibles, il est nécessaire de disposer d'un robot réel. Comme dans le cas du chapitre précédent sur le comportement collectif, cette expérience d'apprentissage et les problèmes qui y sont associés peuvent être correctement analysés seulement s'ils sont menés sur un robot qui fait face aux problèmes dans le monde réel. Les simulations faites par P. Verschure ont eu le mérite d'avoir très bien dégrossi le problème, mais contenaient toujours des éléments irréalistes qui peuvent cacher de gros problèmes. Dans son simulateur, par exemple, la position du robot ainsi

que ses déplacements étaient toujours donnés en coordonnées absolues, ce qui n'a pas de sens pour un robot réel. Ce petit problème n'a heureusement pas causé de problèmes lors de la transposition du simulé au réel, grâce au fait que d'autres mécanismes en contrebalançaient les effets. D'autres cas de transfert simulé-réel ont rencontré ce même problème sans arriver à le résoudre si facilement. La validation sur un système réel reste finalement la seule validation crédible.

5 L'ÉVOLUTION

Si l'apprentissage présenté dans le chapitre précédent est une façon de saisir les éléments importants de l'interaction robot-environnement, cette méthode a, comme on l'a vu, de grandes limitations. Ce nouveau chapitre se propose d'explorer une autre approche d'adaptation, basée sur le principe de l'évolution naturelle des espèces. Trois expériences permettent de comprendre comment cette technique peut être utilisée sur un robot mobile réel, comment on peut affiner cette technique en respectant quelques règles inspirées de la nature et enfin comment on peut associer la technique évolutionniste à celle de l'apprentissage.

5.1 Les algorithmes génétiques

Les algorithmes génétiques (AG) sont généralement considérés comme une méthode d'optimisation d'inspiration biologique [Goldberg89]. Le mécanisme duquel ils s'inspirent est l'évolution darwinienne. Le concept de base est que les individus d'une population qui, dans leur interaction avec l'environnement, sont les mieux adaptés à leur environnement, peuvent se reproduire plus que les individus mal adaptés. Ceci permet à l'espèce de s'adapter à son environnement et ainsi à mieux survivre, se reproduire etc.

Au niveau des AG en tant qu'outil d'optimisation mathématique, ce concept a été réduit pour en faire un algorithme permettant de trouver des bonnes solutions (les individus du concept évolutif) à un problème mathématique (l'environnement du concept évolutif) selon une fonction d'évaluation donnée (le degré d'adaptation du concept évolutif). Comme dans le modèle naturel, les AG sont basés sur le génotype, qui contient l'information nécessaire à la création de l'individu et permet la transmission de cette information d'une génération à l'autre. Dans ce génotype on décrit, sous une forme linéaire de chaîne d'éléments d'information, les paramètres des solutions au problème posé. Pour démarrer le processus de recherche, on génère une population de génotypes aléatoires (figure 64). Chaque génotype est ensuite décodé pour créer un phénotype, qui est une proposition de solution au problème, caractérisée par les paramètres contenus dans son génotype. Chaque solution est ensuite testée, et son aptitude à résoudre le problème est quantifiée dans une note basée sur une fonction d'évaluation (appelée aussi *fitness*). Une fois récoltées les notes d'évaluation de toute la population, on procède à la reproduction sélective de cette population pour créer une nouvelle population de génotypes, considérée comme une nouvelle génération. Le type de sélection utilisé pour cette reproduction peut varier, mais est toujours basé sur un avantage donné aux solutions ayant reçu une bonne note d'évaluation. On peut ainsi appliquer des méthodes élitistes très strictes, comme prendre les 20% meilleures solutions et les reproduire chacune 5 fois. Les méthodes plus utilisées laissent plus de place au hasard, en donnant par exemple une meilleure probabilité de reproduction aux solutions les meilleures, sans en garantir la reproduction ni bloquer les possibilités de reproduction des solutions moins bonnes. Une fois la reproduction effectuée, on applique deux autres opérateurs génétiques sur la nouvelle population de génotypes: Le croisement et la mutation. Le croisement tente de

reproduire l'effet du croisement naturel, en coupant deux génotypes à un certain endroit et en échangeant, entre les deux génotypes, une des deux parties ainsi formées. La mutation change de façon aléatoire un paramètre d'un génotype choisi aléatoirement. Ces deux opérateurs sont utilisés pour permettre une meilleure recherche des solutions dans l'espace de recherche, par le mélange/partage des paramètres trouvés et par un bruit aléatoire. La population ainsi créée est reinjectée dans l'évaluation et le cycle est répété jusqu'à ce que le résultat de l'évaluation est considéré comme suffisamment bon.

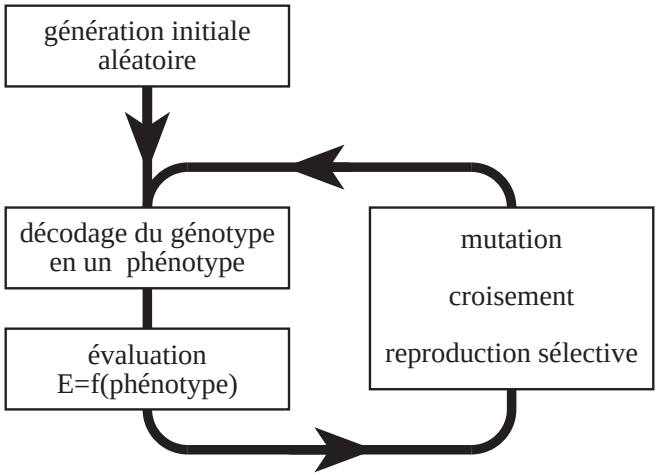


Figure 64: Boucle de fonctionnement des algorithmes génétiques classiques

Les caractéristiques des algorithmes génétiques par rapport à d'autres méthodes d'optimisation sont les suivantes:

- Ils travaillent sur une population de solutions et non pas sur une seule.
- Seulement la valeur ponctuelle de la fonction à optimiser est prise en compte, et non pas sa dérivée ou d'autres valeurs annexes.
- Le codage des paramètres dans le génotype est complètement découplé du problème et de la nature des solutions.
- Le fonctionnement de base est probabiliste et non déterministe.

Ces caractéristiques font que les AG sont utilisés pour résoudre des problèmes complexes, difficiles à résoudre par les méthodes classiques à cause, par exemple, de la forme de l'espace des solutions. Si d'un côté ils arrivent à résoudre des problèmes complexes, les AG ont le désavantage d'être lents et permettent de trouver des solutions proches de l'optimum, mais ne garantissent rien quant à leur convergence et à la qualité de la solution trouvée.

5.2 Les algorithmes génétiques, les réseaux de neurones et les robots mobiles

Dans le cadre des robots mobiles autonomes, l'un des principaux problèmes, comme nous l'avons vu, est de créer un système bien adapté à l'environnement dans lequel il opère et bien adapté à la tâche à exécuter. Or ce problème, par sa nature, par le

type d'environnement auquel il fait face et par le genre de comportements qu'il nécessite, est très similaire au problème d'adaptation des animaux dans des environnements naturels. Les concepts originaux de l'évolution darwinienne comme ils ont étés décrits plus haut sont ainsi parfaitement adaptés. Il est donc intéressant de retourner à l'origine des algorithmes génétiques et appliquer la méthode évolutionniste en gardant l'aspect d'adaptation à l'environnement, tout en appliquant les opérateurs génétiques nécessaires dans un système technique tel qu'un robot (figure 65).

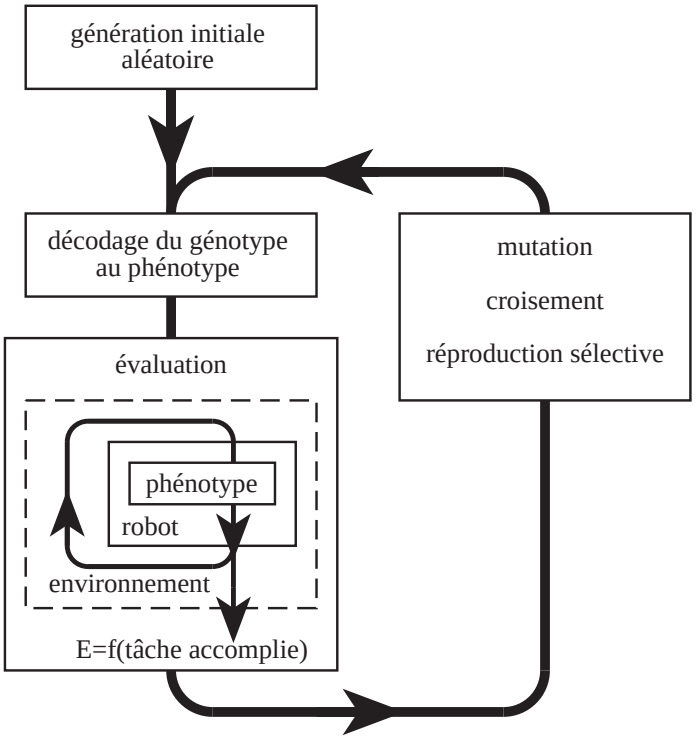


Figure 65: Méthode d'application des AG aux robots mobiles réels

La fonction d'évaluation est, dans ce cas, une mesure de la performance du robot à travers sa vie. Cette mesure permet d'éviter des descriptions trop précises de la tâche, laissant une plus grande autonomie au robot. Les problématiques du monde réel, des représentations internes, de la cohérence robot-système de contrôle-environnement sont prises en charge par l'évolution grâce à l'interaction continue entre tous ces éléments. C'est cette interaction qui crée, au fil des générations, la structure de contrôle la mieux adaptée. Dans ce sens la méthode évolutionniste dépasse le simple rôle de méthode d'optimisation, en prenant en main le développement d'un système de contrôle.

L'interaction robot-environnement joue le rôle déterminant de canalisatrice du développement, et doit être particulièrement soignée. C'est pour cette raison que la contrainte d'avoir un robot réel dans un monde réel a été encore plus prise en considération lors des expériences suivantes.

Le phénotype que nous avons utilisé est le réseau de neurones artificiel. Ce type de structure se prête très bien à l'application des AG car sa structure est d'une granularité assez fine pour permettre une bonne adaptabilité du système. De plus une petite variation

au niveau de la structure ou des valeurs synaptiques des réseaux de neurones provoque une petite variation au niveau comportemental du système, du moins pour des réseaux sans rebouclements internes. Cette continuité est très utile pour garantir un bon fonctionnement des algorithmes génétiques.

5.3 Un premier test de faisabilité

La première expérience réunissant AG et réseaux de neurones sur un robot mobile réel devait répondre à la question de base: Est-il possible d'appliquer des algorithmes génétiques sur un robot réel? Par la suite il s'agissait d'observer les attitudes de l'AG à saisir l'interaction robot-environnement et à l'exploiter au mieux.

Afin d'avoir une large palette de comparaisons, la tâche visée dans cette première expérience a été l'évitement d'obstacles.

5.3.1 Description de l'expérience

Nous avons utilisé le robot Khepera, décrit dans la section 2.3 *L'outil Khepera*. Le robot a été placé dans l'environnement décrit en figure 66. Cet environnement comporte des obstacles qui sont tous du même type, mais qui créent des configurations géométriques très variées. L'environnement a été volontairement fait en forme de couloir circulaire, de façon à avoir un chemin assez bien défini.

Le robot a été connecté à une station de travail SUN, selon la figure 19 à la page 45. Nous avons choisi de garder les processus nécessitant du temps réel strict (contrôle des moteurs, échantillonnage des capteurs etc) sur le robot, mais d'exécuter la partie de contrôle et de gestion de l'algorithme génétique sur la station de travail. Grâce aux grandes constantes de temps en jeu (supérieures à 20 ms) il est possible de contrôler sans problèmes le comportement du robot à travers la ligne sérielle.

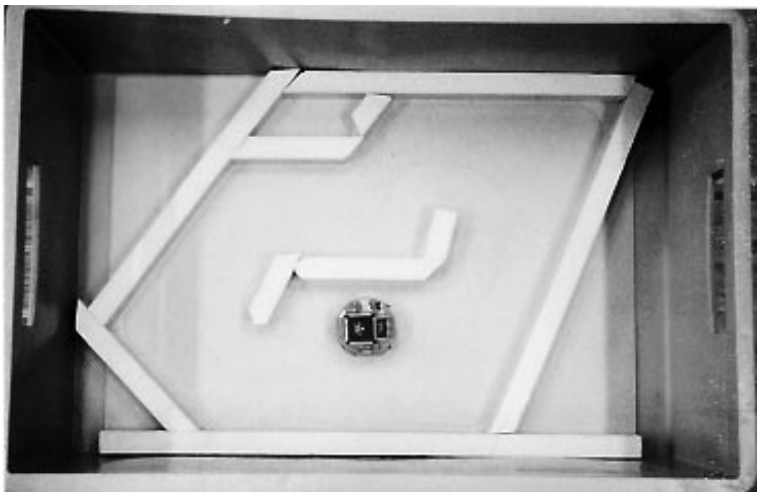


Figure 66: Environnement (50x80 cm) dans lequel le robot a été plongé lors du premier test

Le réseau de neurones utilisé dans ce problème est basé sur les expériences précédentes dans l'évitement d'obstacles (chapitre 3 *L'interaction avec l'environnement*) faites sur la base du véhicule de Braitenberg de type 3c (figure 24). Ces expériences avaient été faites avec une structure d'interconnexions directes entre capteurs et moteurs comme illustré en figure 67a. Les entrées reçoivent directement les valeurs normalisées des capteurs, et les sorties commandent des régulateurs en vitesse qui contrôlent la vitesse de rotation des roues. Ce réseau de neurones simplifié (uniquement une somme sans fonction non-linéaire à la sortie) est un des systèmes les plus performant et robustes que l'on connaisse pour réaliser le comportement d'évitement d'obstacles avec les capteurs de proximité présents sur le robot Khepera.

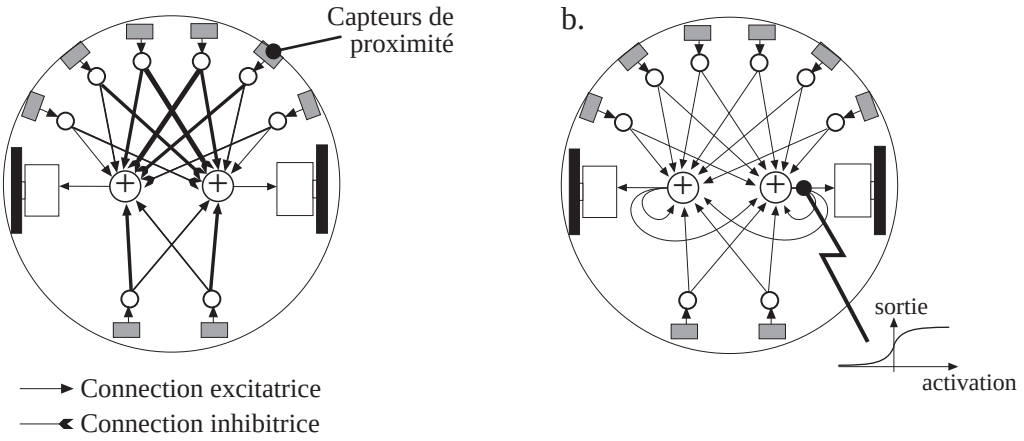


Figure 67: Structures des réseaux (a) basé sur le véhicule 3c de Braitenberg, et adapté pour faire de l'évitement d'obstacles (b) soumise à l'algorithme génétique

La structure de base utilisée pour l'évolution par algorithme génétique est présentée dans la figure 67b. Elle est composée par une connectivité totale entre capteurs et moteurs, similaire à celle du véhicule de Braitenberg, à l'exception des connexions récurrentes et des fonctions sigmoïdales à la sortie des neurones. Ces deux améliorations ont été faites pour améliorer les possibilités du réseau, surtout afin de pouvoir faire face à des cul-de-sac étroits et symétriques. Dans ce dernier cas, en effet, le véhicule de Braitenberg, en ayant une structure symétrique, n'arrive pas à trouver une voie de sortie.

La structure du réseau étant fixe, l'évolution de ce réseau a porté uniquement sur les poids des connexions. Le génotype contient, à la suite, tous les poids des connections du réseau. Ces nombres sont considérés comme des cellules indivisibles, qui ne peuvent pas être coupée pendant le croisement, par exemple, et dont la mutation se fait par addition d'un nombre aléatoire.

Pour la population de 100 individus soumise à l'algorithme génétique, chaque génotype a été décodé pour générer un réseau neuronal. Le réseau ainsi crée a été testé sur le robot connecté à la station de travail: Les valeurs des capteurs ont été introduites à l'entrée du réseau et les sorties du réseau ont servi de consignes aux contrôleurs de vitesse des moteurs. Les valeurs des capteurs et des moteurs étaient mises à jour toutes les 330ms. La durée du test était de 20 secondes. Pendant toute la durée du test et à cha-

que mise à jour du réseau, la vitesse des moteurs ainsi que l'activation des capteurs étaient mesurée. La fonction d'évaluation du robot était calculée en sommant à chaque itération la valeur issue de l'équation suivante:

$$\Phi = V \cdot (1 - \sqrt{\Delta V}) \cdot (1 - i) \qquad \text{(Eq. 12)}$$

Dans laquelle:

- V est la vitesse moyenne des deux moteurs en valeur absolue
- ΔV est la valeur absolue de la différence des vitesses des deux moteurs
- i est la valeur maximale normalisée des 8 capteurs de proximité

Dans cette fonction d'évaluation le premier terme récompense la vitesse du robot, le deuxième récompense une trajectoire droite et le troisième l'éloignement des obstacles. Le deuxième terme, qui est le moins explicite, est aussi celui qui a nécessité le plus grand effort de recherche. Comme discuté plus loin, ce terme a été affiné par des expériences successives. Finalement la fonction racine carrée est celle qui a mieux permis de pénaliser fortement des faibles virages (différences de vitesses normalisées entre 0 et 1) tout en garantissant une bonne dynamique, comme le montre la figure 68.

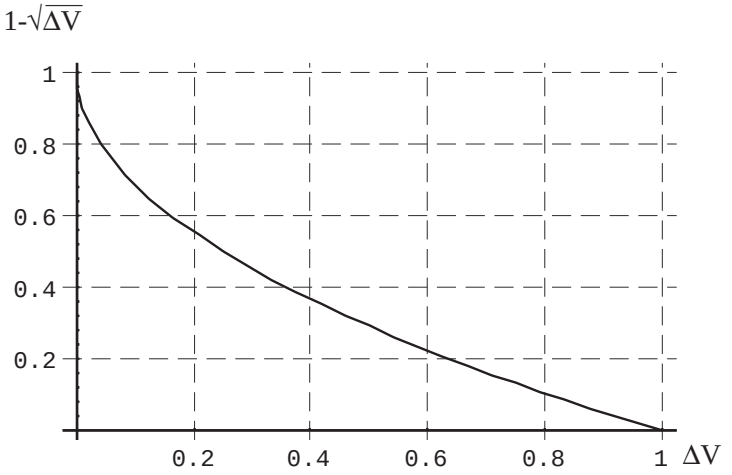


Figure 68: Graphique illustrant l'effet du deuxième terme de la fonction d'évaluation. La fonction racine carrée permet de pénaliser fortement des faibles différences de vitesse

Cette fonction d'évaluation a été calculée pour chaque mise à jour du réseau (toutes les 330 ms) et moyennée sur toute la durée de la vie de l'individu. Entre le test de deux individus successifs, le robot a été déplacé aléatoirement pendant 5 secondes afin d'éviter trop d'influences du comportement d'un individu sur les chances du suivant.

L'algorithme génétique appliqué dans cette expérience est basé sur le logiciel décrit dans [Floreano93] et suit très fidèlement l'algorithme et les paramètres décrits dans [Goldberg89], avec *fitness scaling* et *roulette wheel selection*, *biased mutation* [Montana89] et croisement en un point.

5.3.2 Résultats

Comme le montre la courbe des notes d'évaluation (figure 69), le comportement du robot s'améliore au fil des générations.

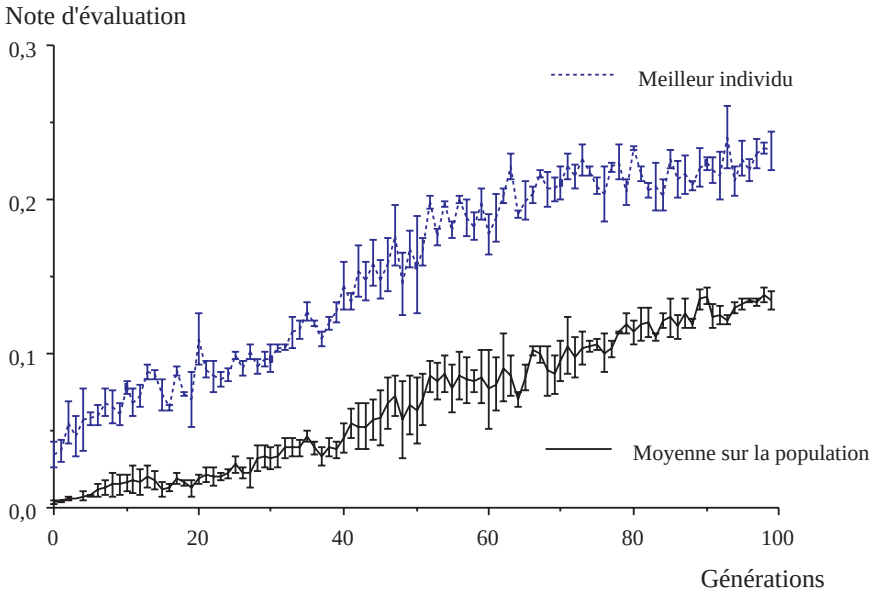


Figure 69: Courbes de la note d'évaluation moyenne sur toute la population et de celle du meilleur individu pour les premières 100 générations. Ces données sont le résultat d'une moyenne sur trois essais

Une analyse plus détaillée de l'évolution, illustrée dans la figure 70, montre qu'au début les meilleurs individus se déplacent de façon rectiligne, mais lentement. Ils ont par contre de la peine à discriminer correctement entre un obstacle et de l'espace libre, et ils restent bloqués contre un mur de façon aléatoire, selon leur position initiale dans l'environnement.

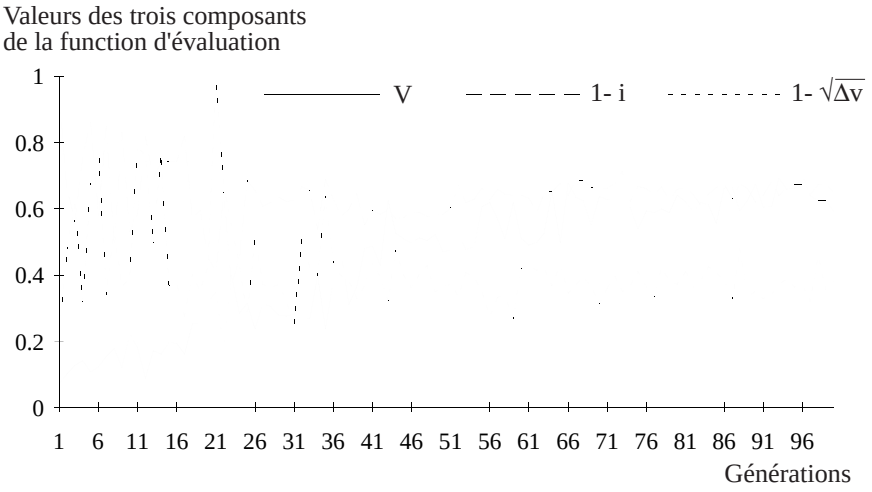


Figure 70: Evolution des trois composantes de la fonction d'évaluation au long des générations

Une autre tendance lors des premières générations est celle de tourner sur place. C'est tout au long des 30 premières générations que se construit le comportement d'aller droit et d'éviter les obstacles. Après la trentième génération l'amélioration est essentiellement due à une augmentation de la vitesse d'avancement, qui ne dépasse toutefois pas les 50 mm/s, par rapport aux 80 mm/s que permet la structure de contrôle. Le comportement final est un évitement d'obstacles très robuste qui a été testé avec le même succès dans d'autres environnements.

5.3.3 Discussion

Remarquons d'abord que l'algorithme génétique est arrivé à résoudre le problème de la recherche des poids du réseau défini plus haut, pour arriver à générer un comportement d'évitement d'obstacles. Ce résultat prouve la faisabilité de l'approche et donne un bon résultat, comparable à d'autres méthodes comme l'apprentissage par renforcement [Touzet96] ou par auto-organisation [Gaussier94b]. En plus de cela, le comportement généré arrive à résoudre des situations que le véhicule de Braitenberg n'arrivait pas à résoudre. Le véhicule de Braitenberg, ayant une structure symétrique et sans mémoire, n'arrive pas à sortir le robot de situations symétriques comme celle de la figure 71a, où il reste piégé dans un coin. Le réseau généré par l'évolution arrive à éviter le blocage en utilisant les connections récurrentes se trouvant sur les unités de sortie. Ces connections lui permettent de réaliser une mémoire à court terme, suffisante pour faire tourner le robot lors de situations comme celle illustrée dans la figure 71b.

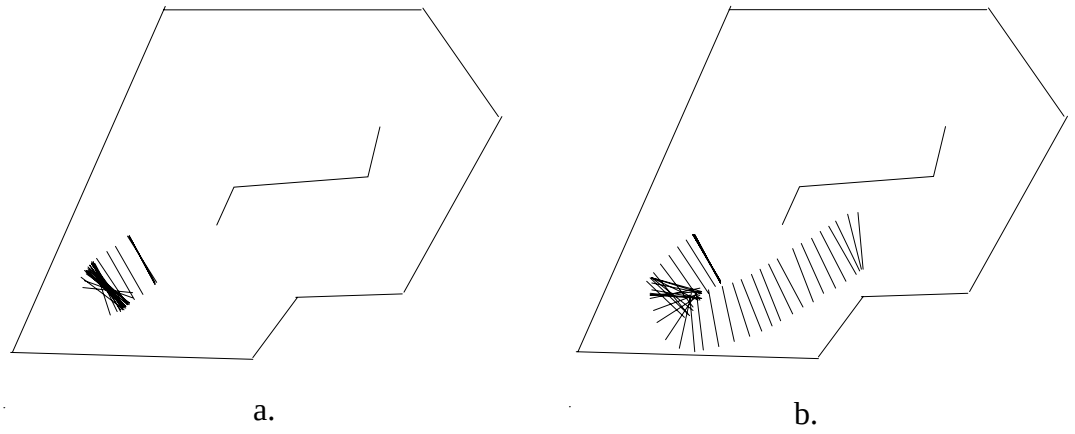


Figure 71: Comparaison du comportement du véhicule de Braitenberg (a) et du robot obtenu par évolution (b). Les traits représentent la position du robot par une ligne reliant les deux roues. Cette position a été mesurée par le système KPS et tracée toutes les 100 ms. On remarque que le véhicule de Braitenberg reste bloqué dans les coins, tandis que le réseau évolué arrive à échapper de cette situation

D'autres aspects intéressants sont plus cachés mais pas moins importants. Si l'on observe les populations finales des différents expériences effectuées, on remarque que tous les individus se déplacent en avant, à savoir du côté duquel se trouvent le plus grand nombre de capteurs (le haut dans la figure 67). Or la fonction d'évaluation ne favorise pas explicitement une direction d'avancement, du fait que toute expression de la vitesse

est faite en valeur absolue. C'est au niveau de l'interaction entre le robot et l'environnement que les robots allant en avant se trouvent avantagés par rapport à ceux qui reculent, grâce au plus grand nombre de capteurs se trouvant à l'avant. Nous pouvons dire que la direction d'avancement émerge de l'interaction entre robot et environnement sous la pression de la fonction d'évaluation. Les AG sont ainsi en mesure de bien saisir et exploiter les caractéristiques de cette interaction.

Si ces quelques résultats sont encourageants, il faut aussi considérer les aspects négatifs de cette expérience. Le premier problème auquel nous avons du faire face est la définition de la fonction d'évaluation. Beaucoup de critiques envers les algorithmes génétiques argumentent que le temps que l'on gagne avec cette méthode par rapport à une programmation classique, est perdu dans la définition très pointue de la fonction d'évaluation. Dans notre expérience il faut admettre que c'était bien le cas. Nous avons en effet commencé nos expériences avec une fonction d'évaluation similaire à celle décrite plus haut, mais sans le deuxième terme qui récompense les trajectoires rectilignes. De plus l'environnement était beaucoup plus ouvert que celui décrit en figure 66. Les premiers résultats étaient des comportements qui consistaient à tourner en cercle, avec un rayon adapté aux caractéristiques de l'environnement, ce qui maximise la vitesse et minimise la collision avec des obstacles. Nous avons ainsi décidé de rétrécir les espaces libres de l'environnement, mais les robots trouvaient toujours de la place pour tourner en rond. Il a donc fallu ajouter un terme qui pénalise ce comportement. Plusieurs essais ont été fait avant d'atteindre la formule décrite plus haut, comportant une racine carrée qui permet de pénaliser fortement les plus petits virages, mais qui n'a aucune raison théorique sinon celle de conduire à un résultat utilisable. Ce problème lié à la fonction d'évaluation est donc réel. Il est essentiellement dû au fait que dans cette expérience le robot n'a aucune autre pression d'évolution que celle créée par la fonction d'évaluation. Pour cette raison, des comportements qui nous semblent logiques (un animal va d'abord en ligne droite), ne le sont pas pour le robot et doivent être spécifiés de façon très détaillée. Le robot n'ayant pas besoin de manger, ni de chercher un partenaire ni rien d'autre, il se contente de faire le minimum pour satisfaire la fonction d'évaluation. Pour ce type de problèmes les critiques faites aux approches basées sur les algorithmes génétiques sont ainsi parfaitement fondées. Pour mieux exploiter les AG il faut approcher les problèmes d'un point de vue plus "écologique", dans le sens d'une meilleure harmonie entre robot, tâche et environnement. Seulement de cette façon on peut arriver à réduire l'aspect d'évolution forcée pour mieux exploiter l'interaction robot-environnement. L'expérience suivante est un exemple allant dans cette direction.

Un deuxième aspect absent dans cette expérience est celui des représentations internes. Le système développé ici est uniquement et parfaitement réactif, avec une mémoire à très court terme. Est-il possible que l'évolution puisse créer les représentation internes nécessaires au fonctionnement du robot? Si oui, sur la base de quelle structure et de quelle façon? L'expérience suivante essaye de répondre aussi à ces questions.

Le troisième aspect nécessaire à un agent autonome et qui n'est pas présent dans cette approche, est l'adaptation pendant la vie. On peut certainement considérer l'ensemble du système évolutif comme adaptatif, mais l'adaptation nécessaire aux systèmes autonomes doit être bien plus rapide et efficace. L'adaptation évolutive seule limite con-

sidérablement les possibilités de faire face à un monde réel, dans lequel les changements sont continus. La dernière expérience s'occupe spécifiquement de cette question.

Finalement un aspect non négligeable dans cette expérience est le temps d'évolution. L'expérience décrite ici (100 générations de 100 individus) a duré 2-3 jours sans arrêt sur un robot. S'il est clair que quelques jours ne sont pas très longs, il faut considérer que la tâche est particulièrement simple. Nous allons voir qu'avec une complexification de la tâche, ce problème devient important et est une limitation majeure de cette approche.

5.4 Amélioration de l'autonomie et de la fonction d'évaluation

Pour pallier aux défaut de la première expérience, nous en avons conçu une deuxième plus écologique, dans laquelle on met l'accent sur les besoins du robot ainsi que le rôle de son interaction avec l'environnement. Ceci a été fait dans le but de réaliser des tâches plus complexes grâce à une motivation venant plus du robot et de son environnement, que de la fonction d'évaluation. Ainsi nous avons repris la fonction d'évaluation de l'expérience précédente, en enlevant le terme artificiel du milieu. On obtient la forme réduite suivante:

$$\Phi = V \cdot (1 - i) \qquad \text{(Eq. 13)}$$

Dans laquelle:

- V est la vitesse moyenne des deux moteurs en valeur absolue
- i est la valeur maximale normalisée des 8 capteurs de proximité

Cette expression pousse l'évolution vers un robot rapide et capable d'éviter les obstacles.

En ce qui concerne les besoins et l'environnement du robot, nous avons simulé l'équivalent d'une batterie qui alimente le robot. Cette batterie est pleine au début de la vie et se décharge de façon linéaire en 20 secondes. Une fois vide elle provoque la mort du robot. Le robot dispose de la valeur de la charge de la batterie en entrée de son réseau de neurones et peut aller la recharger à un endroit précis de l'environnement, situé dans un coin et dont le sol a été peint en noir (figure 72). La recharge se fait instantanément et ramène la batterie, quel que soit son niveau de charge, au niveau plein correspondant à une durée de vie de 20 secondes. La vie totale du robot a été limitée à une minute, afin de ne pas rallonger excessivement l'expérience. Il faut noter que le robot n'a pas intérêt, dans le sens de la fonction d'évaluation, à rester sur la zone de chargement. En effet celle-ci se trouve entre deux murs, ce qui active les capteurs de proximité et rend nulle la valeur rendue par la fonction d'évaluation.

Le robot a été équipé d'un capteur de couleur du sol, qui permet au système de contrôle ainsi qu'au robot de savoir quand il se trouve sur le chargeur. Enfin une source lumineuse a été placée au dessus du chargeur, afin que le robot puisse s'orienter dans l'environnement. Pour la percevoir, deux capteur de proximité sont utilisés aussi comme

capteurs d'intensité lumineuse et leur valeurs sont placées en entrée du réseau de neurones. L'environnement complet est présenté en figure 72.

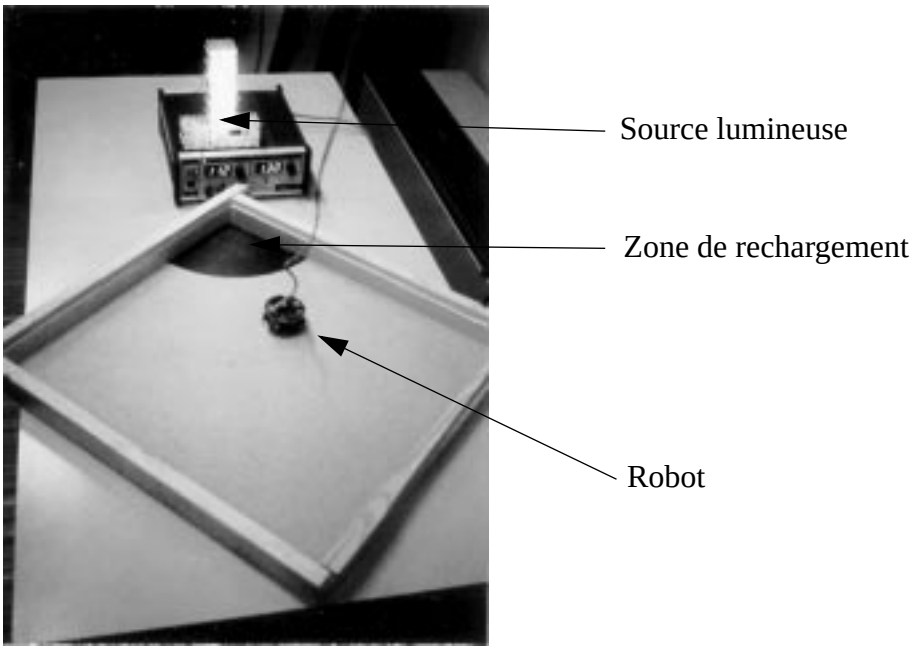


Figure 72: Environnement de la deuxième expérience, d'une taille de 50 x 50 cm

Le réseau de neurones soumis à l'évolution a naturellement aussi été modifié pour pouvoir réaliser la nouvelle tache. Des entrées ont été ajoutées afin de pouvoir introduire les données provenant du capteur de couleur du sol, de la batterie et de la luminosité ambiante. Ensuite il faut considérer que cette nouvelle tâche comporte de la navigation car le robot doit retrouver l'endroit de chargement. Nous avons ainsi arbitrairement décidé de munir le réseau d'une couche caché de 5 neurones, munis de connexions récurrentes. La figure 73 illustre la structure du réseau en tenant compte des adaptations faites.

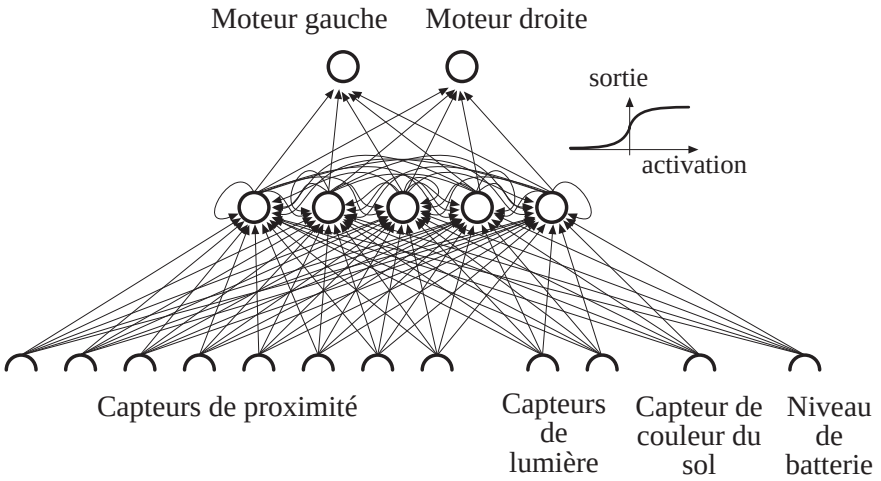


Figure 73: Structure du réseau de neurones utilisé

L'algorithme génétique ainsi que l'installation utilisés lors de cette expérience sont identiques à ceux utilisés lors de l'expérience précédente. Dans ce cas aussi nous avons codé les poids du réseau dans le génotype. La mise à jour du réseau a été faite toutes les 400 ms.

5.4.1 Résultats

La figure 74 montre la croissance régulière des performances des individus au cours des générations.

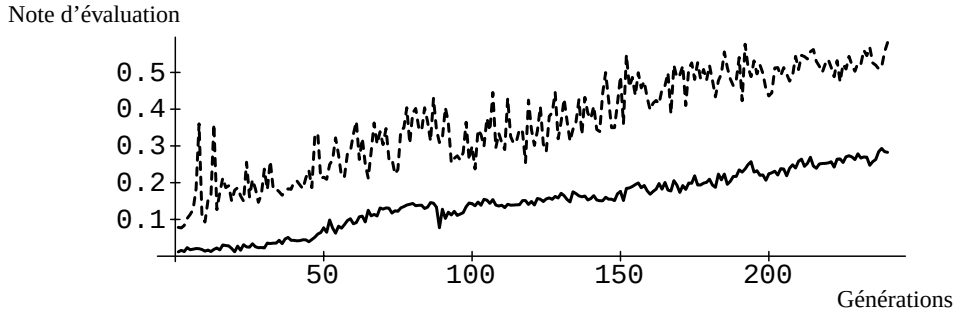


Figure 74: Performance moyenne de la population (trait continu) et du meilleur individu (pointillé) en fonction des générations

Pour pouvoir mieux comprendre le comportement des individus, il est intéressant de voir comment la recharge de la batterie est intégré dans le comportement. La figure 75 illustre la longueur de vie des meilleurs individus au fil des générations. Comme on peut l'observer les premiers individus ne survivent que les 20 premiers secondes d'autosuffisance de base de la batterie, correspondant à 50 mises à jour du réseau, alors qu'après la 200ème génération presque toujours le meilleur individu survit le temps maximum accordé, une minute, correspondant à 150 mises à jour.

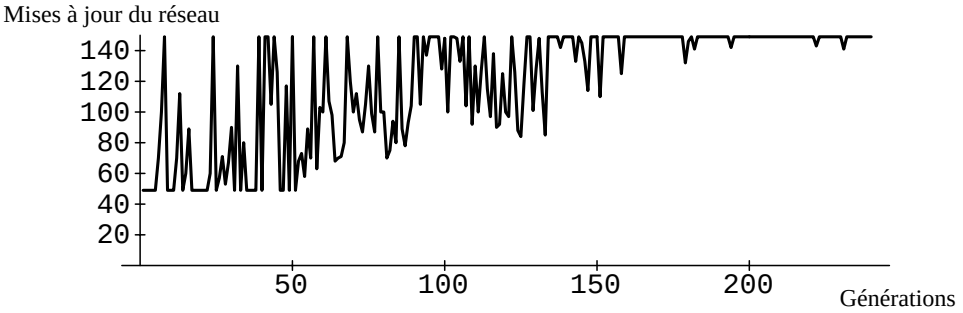


Figure 75: Durée de vie du meilleur individu en fonction de la génération analysée

Lors des premières générations, les robots ont une tendance à améliorer leur performance en suivant les règles dictées par la fonction d'évaluation. Ils ont la tendance à aller vite et à éviter les obstacles. Déjà à partir des premières générations, quelques individus, qui ne sont pas nécessairement les meilleurs du point de vue de la fonction d'évaluation, arrivent à survivre plus longtemps que les autres en allant se recharger. Ces

individus disposent ainsi d'une plus grande durée de vie pendant laquelle il peuvent accumuler des points d'évaluation. La combinaison des capacités de ces individus, parfois moyens mais avec une grande longévité, et des individus efficaces sous le point de vue de la fonction d'évaluation mais incapables de se recharger, porte lentement à la création d'individus bien adaptés. L'adaptation est faite par rapport à la fonction d'évaluation mais en tenant compte des besoins de survie propres au robot. Dans la 240ème génération on arrive à trouver un individu qui arrive à se déplacer dans l'environnement de façon relativement rapide, en évitant les obstacles et en rentrant à la station de chargement approximativement toutes les 19 secondes. Ceci est un comportement proche de l'optimal qui démontre l'affinement opéré par l'évolution.

Analysés de plus près, l'individu se montre capable de rentrer à la station de chargement dans le délai prévu (20 secondes) pour 88 parmi 100 positions et orientations testées dans l'environnement, comme le montre la figure 76. Dans le 12% des cas restant, le robot arrive sur le chargeur dans les secondes suivant le délai fixé. Le réseau de neurones arrive ainsi à s'orienter dans l'environnement et à retrouver le chargeur. Ce fait n'est pas seulement prouvé par le succès du rechargement, mais aussi par le fait que le robot évite soigneusement la zone de recharge s'il a des batteries chargées (figure 77a). Ceci est dû au fait que le robot n'est pas récompensé pour ses actions tant qu'il se trouve sur la station de chargement. Il a donc intérêt à ne pas s'approcher de cette zone tant qu'il en a pas besoin.

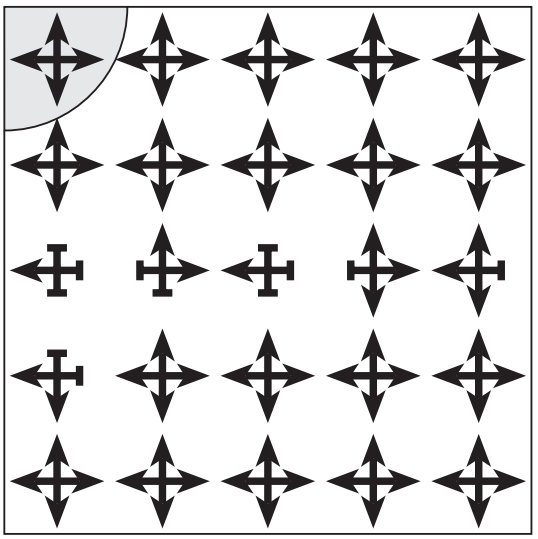
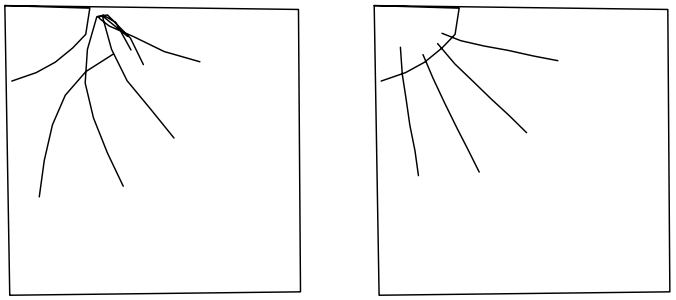


Figure 76: Plan qui illustre la capacité du système à atteindre le chargeur (en haut à gauche) depuis les différentes parties de l'environnement. Les flèches indiquent les positions et les directions depuis lesquelles le robot, parti avec une batterie chargée, arrive à atteindre le chargeur en moins de 20 secondes. Les traits barrés indiquent des positions et orientations depuis lesquelles le robot n'arrive pas à atteindre le chargeur en moins de 20 secondes. Il est à noter que dans ces derniers cas, le robot reste sans batterie très proche du chargeur, la plus grande distance mesurée dans ces cas étant 3 cm.



a. b.

Figure 77: Trajectoire du robot lorsqu'il est positionné face à la zone de chargement (a.) avec la batterie chargée et (b.) avec la batterie presque vide

Nous avons essayé d'analyser le type de représentation interne que le processus d'évolution a formé au niveau du réseau de neurones. Pour faire cela nous avons d'abord analysé l'activité des 5 neurones de la couche cachée pendant l'activité du robot, comme le montre la figure 78.

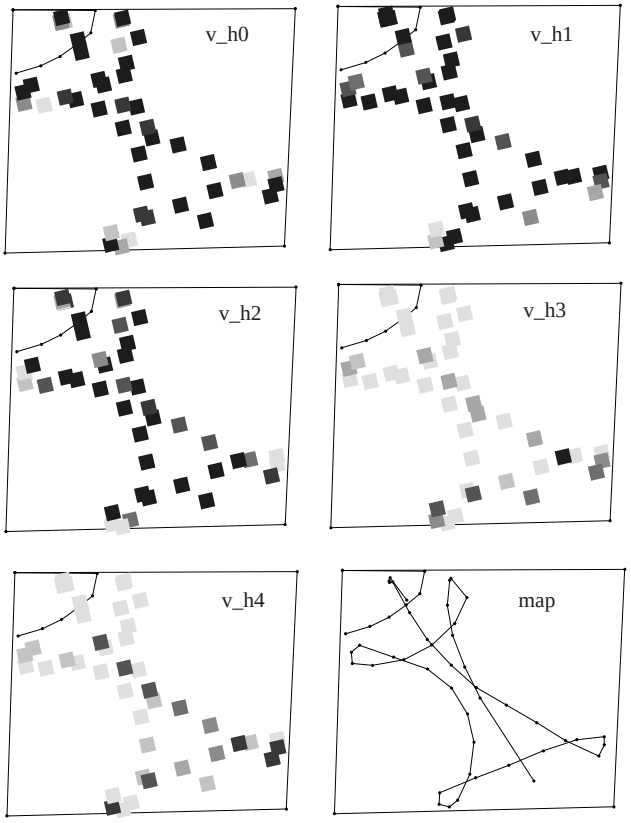


Figure 78: Visualisation, en niveaux de gris, de l'activité des cinq neurones de la couche cachée en fonction de la position du robot dans l'environnement et lors d'une trajectoire typique (dernier graphique)

Ceci a permis d'observer les corrélations entre l'activité des neurones et la situation ou la position dans laquelle se trouve le robot. Les neurones 0, 1 et 2 ont clairement une activité corrélée avec la détection des murs et leur évitement. Le neurone 4 et de façon plus faible le neurone 3, présentent par contre une activité corrélée avec le niveau de charge de la batterie. Le neurone numéro 4, en particulier, devient actif lors de la fin de vie des batteries. La figure 79 illustre l'activité de ce neurone en fonction de la position et l'orientation statique du robot dans l'environnement. Cette activité est dépendante de la position du robot, et permet d'identifier clairement la position du lieu de chargement (en haut à gauche). Or un observateur attentif pourrait argumenter que le fait que la source lumineuse soit au dessus du lieu de chargement facilite la tâche de navigation, la réduisant à un simple suivi de source lumineuse. Pour vérifier la généralité de la représentation interne nous avons placé la source lumineuse dans un autre coin de l'environnement, décalé de 90 degrés par rapport au lieu de chargement, puis laissé continuer l'évolution. Après un nombre très limité de générations (entre 10 et 20) le robot avait retrouvé le lieu de chargement et se comportait comme auparavant, arrivant à se recharger régulièrement toutes les 19 secondes. Au niveau des neurones de la couche cachée on a pu observer une activité similaire à celle présentée en figure 79, mais tournée de 90 degrés. Cette activité a toutefois été observée sur le neurone 3. Ce test, répété et réussi pour les quatre coins de l'environnement, prouve que la représentation interne est relativement générale est peut facilement être adaptée à des nouvelles configurations.

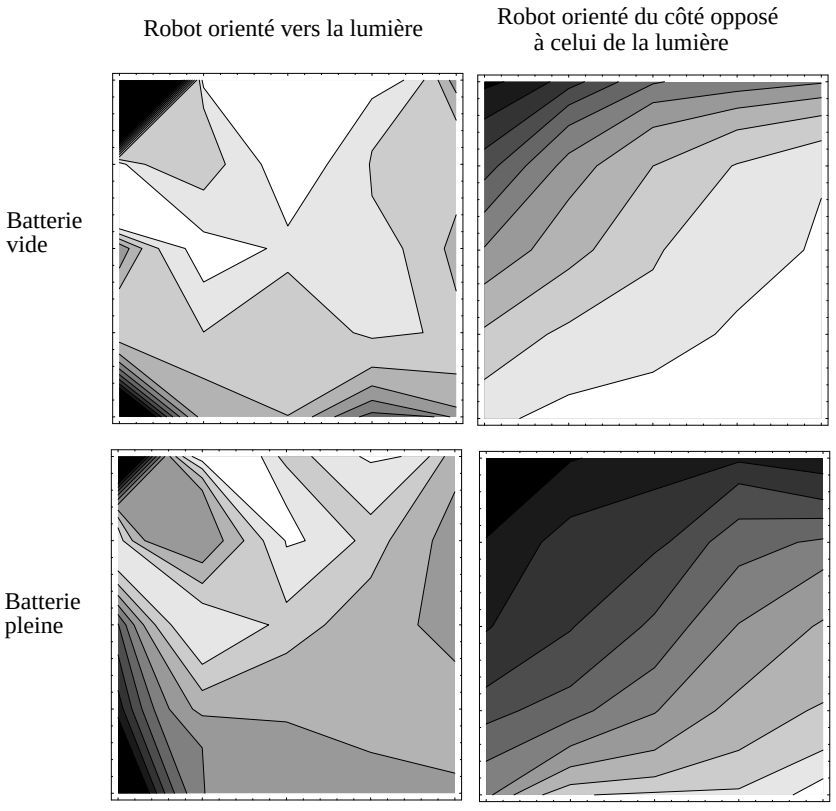


Figure 79: Activité du quatrième neurone de la couche cachée en fonction de la position du robot dans l'environnement

Le processus de navigation proprement dit, qui consiste dans l'utilisation de la représentation décrite ci-dessus pour diriger le robot, n'a pas été étudiée en détail à cause de la complexité des interactions présentes dans le réseau. La seule constatation est que le comportement s'appuie sur une trajectoire de base dictée par un mécanisme interne, qui apparaît aussi si la lumière est éteinte, comme le montre la figure 80. Par dessus ce comportement vient se greffer une correction donnée principalement par le quatrième neurone. Cette correction a lieu pendant les derniers gros virages faits lors de la rencontre des murs, et lors de la dernière ligne droite vers le lieu de chargement.

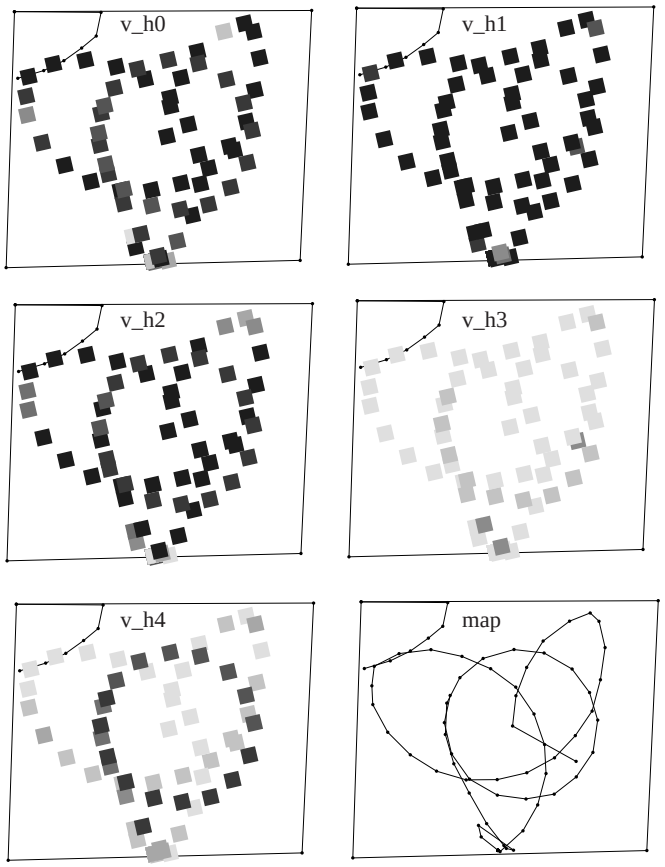


Figure 80: Trajectoire du robot et activation des neurones (représentation de la figure 78) avec la source lumineuse éteinte

5.4.2 Discussion

Cette deuxième expérience montre bien comment la pression de l'environnement et une sorte de métabolisme du robot peuvent diriger l'évolution d'un système soumis aux algorithmes génétiques. Malgré la simplification de la fonction d'évaluation, le comportement développé par le système est nettement plus complexe que le précédent.

La complexité de la tâche de navigation a poussé le système à créer une représentation interne. Les aspects importants dans ce choix sont doubles: D'une part la forme, structure et fonctionnement de cette représentation ont étés modelés sur le problème

même, en tenant compte de l'environnement, du robot et de la tâche à réaliser. Ceci permet d'obtenir une représentation bien adaptée, ce qui simplifie énormément soit la structure de la représentation, soit celle nécessaire à son exploitation. Il n'existe pas d'exemple similaires, à ma connaissance, qui utilisent seulement 5 neurones d'une couche cachée pour réaliser une navigation, même aussi simple que celle-ci. L'imposition d'une structure plus complexe qu'on aurait pu considérer comme utile pour cette tâche, comme par exemple une carte de l'environnement, aurait introduit une complexité bien supérieure à tous les niveaux, tout en étant probablement plus compréhensible lors de la conception ou l'analyse [Gaussier94c].

D'un autre coté, et cela devient encore plus important, l'autonomie du système a été étendue au choix, très controversé, de l'introduction ou pas d'une représentation interne. Le choix fait normalement par les opposants ou les soutenant du besoin de représentations internes est une claire limitation de l'autonomie du robot. Dans notre expérience, le système a choisi de façon autonome (à la structure du réseau près) de créer et utiliser cette représentation interne, alors que son utilité peut même être mise en doute.

Finalement, la seule contrainte fixée dans la fonction d'évaluation est très intimement liée à la tâche que l'on veut voir réalisée par le robot, sans tenir compte des besoins propres ou intermédiaires du robot pour la réaliser. La fonction d'évaluation doit être considérée comme un moyen pour décrire le but à atteindre, et non pas la tâche à exécuter, comme nous l'avions fait dans la première expérience.

Si ces résultats positifs sont parfaitement dans la ligne du développement de robot autonomes et sont prometteurs, un aspect négatif de l'expérience remet en question toute l'approche. Le développement, comme nous l'avions déjà cité plus haut, prend beaucoup de temps. Dans le cas de cette deuxième expérience, il a fallu environ trois semaines de tests continus pour réaliser l'évolution, sans tenir compte des essais fait pour mettre au point l'expérience, ce qui a pris des mois d'expérimentation continue. Ce fait montre bien l'augmentation du temps nécessaire par rapport à l'augmentation de la complexité de la tâche. Une simple extrapolation permet d'affirmer que cette méthode n'est pas applicable dans cette forme pour des tâches d'une complexité plus importante.

Finalement l'adaptation pendant la vie de l'individu n'est pas présente dans cette expérience. Cet aspect est le sujet principal de l'expérience suivante.

5.5 L'apprentissage et l'évolution

Si l'évolution permet de concevoir un système de contrôle en lui laissant une autonomie importante tout en le façonnant sur la base de l'interaction robot-environnement, cette adaptation a lieu uniquement au fil des générations. Si l'environnement subit un changement, comme dans le cas du déplacement de la source lumineuse dans la dernière expérience, le système nécessite le passage à travers plusieurs générations avant de retrouver un comportement correct. La structure de contrôle de l'individu n'a en effet aucune flexibilité au cours de la vie. Or, si l'adaptation au cours des générations est utile et efficace, il est nécessaire de doter l'individu d'une possibilité d'adaptation au cours de sa vie, l'apprentissage.

L'apprentissage n'est pas seulement souhaitable pour améliorer les capacités d'adaptation de l'individu pendant sa vie, mais peut avoir une influence intéressante sur le processus d'évolution. Même dans le cas de non héritage des connaissances accumulées par les parents, plusieurs chercheurs ont en effet mis en évidence que la combinaison de méthodes d'apprentissage pendant la vie et l'évolution permettent d'obtenir des résultats meilleurs que ceux obtenus par l'utilisation d'une seule de ces deux techniques. Nolfi et al. [Nolfi90] ont montrés que la combinaison de méthodes de rétropropagation du gradient et d'évolution appliquées à des agents autonomes permettent d'obtenir des meilleurs résultats qu'avec l'application de l'évolution seule.

Enfin l'utilisation des réseaux de neurones, qui se montrent très bien adaptés à l'application des algorithmes génétiques, se prête très bien aussi à l'application de techniques d'apprentissage. Il serait intéressant de profiter aussi de ces possibilités.

5.5.1 Les types d'apprentissage et de neurones considérés

Comme dans l'expérience sur l'apprentissage (chapitre 4), nous nous sommes intéressés à des règles simples, locales et non supervisées. Le choix a porté encore une fois sur des règles d'apprentissage hebbiennes. Afin de rendre les données de l'expérience le plus proche possibles de la biologie et pour pouvoir appliquer les algorithmes génétiques, un certain nombre de choix ont dû être fait.

Premièrement il n'y a pas de consensus concernant la prédominance d'une règle d'apprentissage en biologie. Vu la diversité des neurones et de leur fonctionnalités, il est probable que des synapses différentes se basent sur des mécanismes d'adaptation différents. Malgré cela les biologistes semblent avoir observé des mécanismes de renforcement ou d'affaiblissement des connexions de type hebbien, c'est à dire basés sur des correspondances d'activité entre le neurone pré-synaptique et post-synaptique. En essayant de combiner ces deux considérations tout en exploitant la présence de l'évolution, nous avons laissé, pour chaque synapse, le choix entre quatre règles d'apprentissage de type hebbien, à savoir pure, post-synaptique, pré-synaptique et basée sur la covariance. Ces règles ont été modifiées afin d'obtenir quelques caractéristiques communes:

- Le poids synaptique ne peut pas croître indéfiniment, comme dans le cas de la plupart des algorithmes d'apprentissage. Dans le cas de cette expérience, la valeur des poids synaptique a été limitée entre 0 et 1. Cette caractéristique semble plus réaliste que l'accroissement illimité (comme dans la rétropropagation du gradient, par exemple). Cette limitation est obtenue par l'introduction dans les règles d'apprentissage d'un terme qui prend en compte le poids synaptique précédant l'adaptation.
- Contrairement aux règles d'apprentissage courantes, les règles d'adaptation utilisées ici ne peuvent pas changer le signe des poids. Cette caractéristique de la connexion (inhibitrice ou excitatrice) est définie génétiquement et ne peut pas être changée pendant la vie. Ce choix est motivé par l'existence de cette caractéristique en nature: Une connexion neuronale biologique est de type excitateur ou inhibiteur mais ne change pas de type au cours de l'apprentissage. Cette caractéristique donne

d'ailleurs une meilleure stabilité au système.

Les règles ont pris la forme suivante:

1 La règle hebbienne pure a été modifiée pour satisfaire la contrainte de limitation de la valeur du poids synaptique:

$$\Delta w_{ij} = (1 - w_{ij}) x_i \; y_j \qquad \text{(Eq. 14)}$$

Avec x_i représentant l'activation du neurone pré-synaptique i, y_j l'activation du neurone post-synaptique j, w_{ij} le poids de la connexion entre les neurones i et j, et Δw_{ij} la modification des poids.

2 La règle hebbienne postsynaptique a aussi été modifiée:

$$\Delta w_{ij} = w_{ij}(-1+x_i)\; y_j + (1 - w_{ij})\; x_i \; y_j \qquad \text{(Eq. 15)}$$

3 La règle hebbienne présynaptique prend la forme symétrique:

$$\Delta w_{ij} = w_{ij}(-1+y_j)\; x_i + (1 - w_{ij})\; x_i \; y_j \qquad \text{(Eq. 16)}$$

4 Enfin, la règle de la covariance, ou règle de Hopfield [Hopfield82], détecte et agit en fonction de la différence d'activité entre neurone pré- et post-synaptique. Elle a aussi été modifiée pour accroître le poids synaptique si la différence entre les deux activations est inférieure à l'activation moyenne d'un neurone (0.5), et le diminuer si la différence est supérieure à cette valeur. Cette valeur de 0.5 est due aux fonctionnalités des neurones moteurs, pour lesquels les vitesses des moteurs (positives et négatives) correspondent à une activité neuronale allant de 0 à 1, ce qui signifie qu'une activité neuronale de 0.5 correspond à une vitesse nulle.

$$\Delta w_{ij} = \begin{cases} (1 - w_{ij}) F(x_i, y_j) & \text{si } F(x_i, y_j) > 0 \\ w_{ij}\, F(x_i, y_j) & \text{autrement} \end{cases} \qquad \text{(Eq. 17)}$$

Avec $F(x_i, y_j)$ étant la fonction sigmoïdale avec la forme suivante:

$$F(x_i, y_j) = \tanh(4|x_i - y_j| - 2) \qquad \text{(Eq. 18)}$$

$F(x_i, y_j) > 0$ si la différence des deux activations est supérieure ou égale à 0.5, $F(x_i, y_j) < 0$ si la différence est inférieure à 0.5.

La mise à jour de la synapse est aussi légèrement corrigée par rapport à (Eq. 5):

$$w_{ij}^{t} = w_{ij}^{t-1} + \eta \Delta w_{ij}^{t} \qquad \text{(Eq. 19)}$$

Dans laquelle w_{ij}^{t} est le poids de la connexion entre le neurone présynaptique i et

le neurone post-synaptique j au temps t, w_{ij}^{t-1} est le même poids au temps (t-1), Δw_{ij}^{t} est la modification des poids au temps t et nous avons ajouté η qui représente le gain d'apprentissage. Ce gain, déterminé par l'évolution et dont la description se trouve dans le génotype, permet, si mis à 0, d'annuler l'apprentissage sur une synapse.

Un autre choix a porté sur la fonction d'activation des neurones. Dans les structures neuronales biologiques on retrouve une grande variété de types de neurones ainsi que de types d'influence que le signal synaptique peut provoquer dans la cellule postsynaptique. Nous avons voulu ici aussi diversifier les fonctionnalités des neurones en permettant à l'algorithme génétique, pour chaque synapse, de choisir parmi deux types d'effets [Kay94]: Le premier type d'entrée synaptique, utilisé dans les expériences précédentes, active le neurone proportionnellement au signe et à la valeur du signal pondéré. Pour être cohérent avec la terminologie de [Kay94], nous allons appeler ici ce type de synapse *driving*, et noter les poids correspondant $w^{driving}$. L'alternative à ce premier type de synapse est appelé *modulatory* et son poids est noté $w^{modulatory}$. Les synapses de cette dernière catégorie ne peuvent pas générer d'activation dans le neurone postsynaptique, mais peuvent seulement en moduler l'activation existante. Cette modulation n'est pas symétrique par rapport au zéro, mais par rapport à la valeur 0.5. Cela tient compte du fait que, pour les neurones moteurs, la gamme des vitesses des moteurs (positive et négative) correspond à une activité neuronale allant de 0 à 1, ce qui signifie qu'une activité neuronale de 0.5 correspond à une vitesse nulle. La modulation a ainsi été réalisée de façon à ne pas influencer le signe de la vitesse. Ce type de synapse *modulatory* veut simuler les canaux à gain de contrôle dépendant de la tension du soma, tels qu'on les trouve dans les cellules pyramidales [Engel92] [Singer90]. Les synapses qui portent ce type de signaux correspondent normalement aux connexions horizontales à l'intérieur du cortex, et sont modifiables comme les autres synapses.

La fonction d'activation qui permet de considérer ces deux types d'influences synaptique a été prise des travaux de Kay et Phillips [Kay94] et prend la forme suivante:

$$y = \frac{1}{1 + e^{-A(d,m)}} \qquad \text{(Eq. 20)}$$

$$A(d,m) = \frac{d}{2}\,(1 + e^{2\,d\,m}) \qquad \text{(Eq. 21)}$$

$$d = \sum_{j=1}^{n} x\ w_j^{driving} \ , \ m = \sum_{j=1}^{n} x\ w_j^{modulatory} \qquad \text{(Eq. 22)}$$

Dans laquelle y est la sortie du neurone, d est la somme des poids pondérés des signaux provenant des neurones présynaptiques ayant une synapse de type *driving* et m est la somme des poids pondérés des signaux provenant des neurones présynaptiques ayant une synapse de type *modulatory.*

L'influence des deux types de synapse sur le signal de sortie du neurone est illustré dans la figure 81. Si la somme des signaux modulateurs est égale à zéro, l'équation (Eq. 20) correspond à une fonction sigmoïdale par rapport aux entrées de type *driving*. Si le signal modulateur est positif, lorsque le neurone est activé positivement par les entrées *driving*, l'activité est accentuée. De même si le signal modulateur et l'activité du neurone sont les deux négatifs. Dans le cas où le signal modulateur et l'activité du neurone sont de signe contraire, la sortie est atténuée.

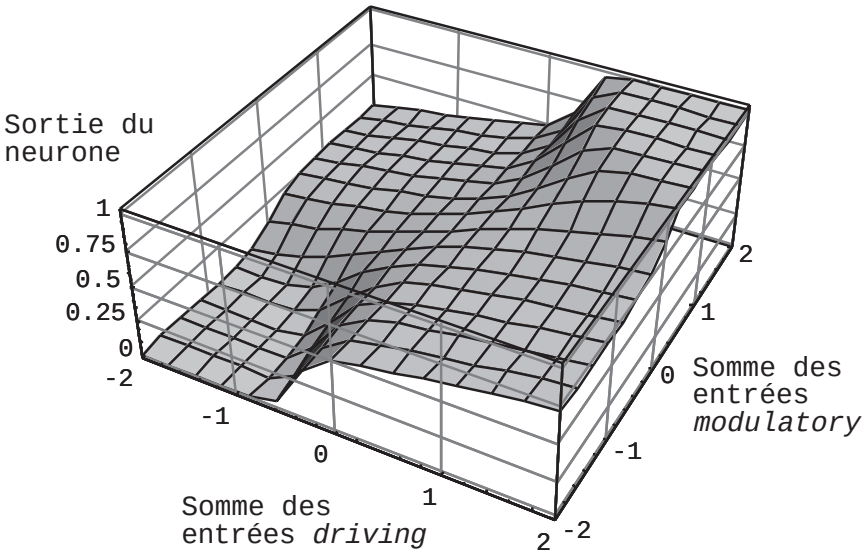


Figure 81: Activation en fonction des sommes pondérées des entrées *driving* et *modulatory*

5.5.2 Description de l'expérience et du réseau utilisé

Cette expérience est aussi un essai qui doit pouvoir être comparé avec d'autres expériences similaires. Ainsi nous avons repris l'expérience d'évitement d'obstacles. L'environnement et le but visé sont donc identiques à ceux décrits dans la section 5.3.

Le réseau complet utilisé dans cette expérience a une structure fixe similaire à celle décrite dans la section 5.3 (page 102), à savoir huit entrées prenant la valeur des huit capteurs de proximité, une couche cachée d'un seul neurone ayant une connection récurrente et deux neurones de sortie dont l'activation commande la vitesse des deux moteurs (figure 82a). De plus, des connections directes relient les entrées aux neurones de sortie. Le neurone de la couche cachée remplace les connexions récurrentes sur les neurones de sortie qui étaient présentes dans l'expérience de la section 5.3. Cette dernière modification a été introduite afin de permettre une meilleure analyse des fonctionnalités associées aux connections récurrentes. Les poids synaptiques de ce réseau peuvent être représentées sous forme d'une matrice des connections, comme le montre la figure 82b. En entrée de cette matrice se trouvent les entrées du réseau ainsi que la sortie du neurone de la couche cachée au temps *t*. En sortie on obtient les activations des neurones de sortie ainsi que l'activation du neurones de la couche cachée au temps *t+1*.

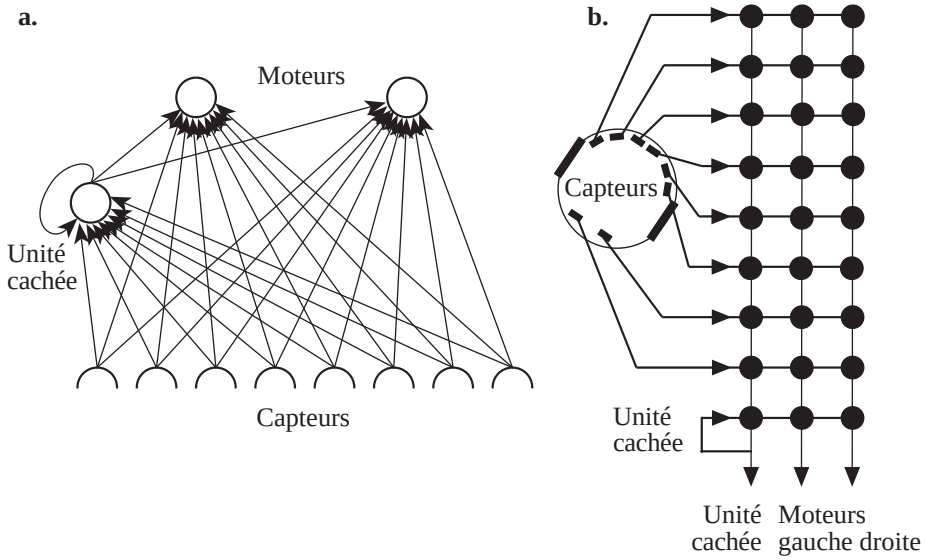


Figure 82: a. Structure du réseau avec une couche cachée de un neurone. b. Représentation du réseau sous forme de matrice de synapses

5.5.3 Le codage du génotype

Le génotype code les caractéristiques individuelles de chacune des 27 synapses de la matrice présentée dans la figure 82b.

Comme nous l'avons vu dans la section précédente, chaque synapse est caractérisées par quatre propriétés, à savoir le type de connexion (inhibitrice ou excitatrice), le type d'influence sur le neurone postsynaptique (driving ou modulatory), la règle d'apprentissage appliquée (au choix parmi les quatre présentées dans la section 5.5.1) et la valeur du gain d'apprentissage η (Eq. 19). Ces quatre paramètres sont codés sur 6 bits comme le montre la figure 83. Ce codage ne permet pas de changement de la structure du réseau.

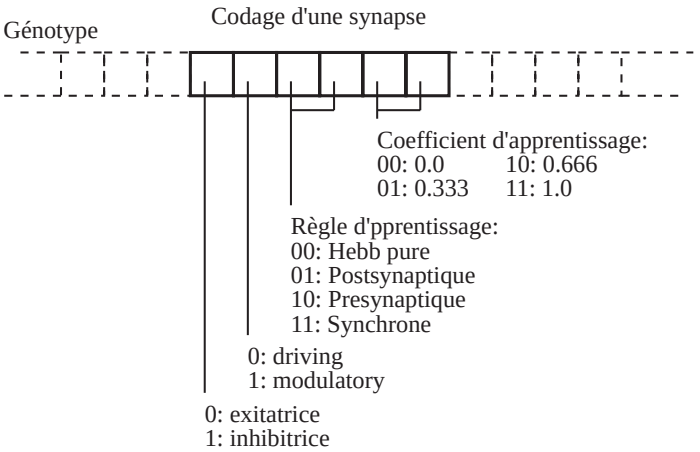


Figure 83: Codage des caractéristiques d'une synapse dans le génotype

Lors du décodage du génotype, on crée le réseau de neurones correspondant et on initialise les poids des connexions avec des valeurs petites (entre 0 et 0.1) aléatoires. Comme dans l'expérience de la section 5.3, l'activité du réseau est calculée toutes les 330ms. Dans cette expérience on effectue en même temps la mise à jour des valeurs synaptiques.

5.5.4 Résultats

En environ 50 générations l'algorithme génétique a développé un individu capable d'effectuer de l'évitement d'obstacles. Ce comportement ne dépend pas des poids synaptiques initiaux du réseau et est développé dans les 6 premiers échantillonnages du système (moins que 2 secondes). Le comportement d'évitement d'obstacles est affiné dans les 10-15 secondes qui suivent cet apprentissage rapide. Après 20 secondes d'apprentissage, le comportement ne change pratiquement plus.

Généralement, le comportement qui a été développé est différent de celui rencontré dans l'expérience décrit dans la section 5.3. Le robot effectue en effet une sorte de suivi de mur, au lieu de l'évitement pur et brusque rencontré auparavant. Dans une première phase le robot cherche une parois, qu'il garde à sa droite et suit à une distance d'environ 2 cm. Une action sur deux consiste à bouger vers la parois pour vérifier la présence du mur. Si le mur est détecté, le robot continue sa trajectoire. Si la parois se montre être longue, le robot accélère et les oscillations de contrôle du mur sont atténuées. Dès qu'un coin est rencontré ou la parois perdue, le robot s'arrête brusquement, fait marche arrière en se tournant dans la bonne direction et repart en suivant la nouvelle parois. Déjà lors du deuxième tour (figure 84b) le comportement est mieux adapté à l'environnement dans lequel le robot se déplace: Les corrections sont moins brusques et la vitesse, généralement plus basse, est légèrement augmentée lors des longues trajectoires droites. La figure 84 illustre la trajectoire du robot lors du premier et du deuxième tour de l'environnement. La vitesse atteint les mêmes valeurs que celles mesurées dans la première expérience comportant les algorithmes génétiques (section 5.3).

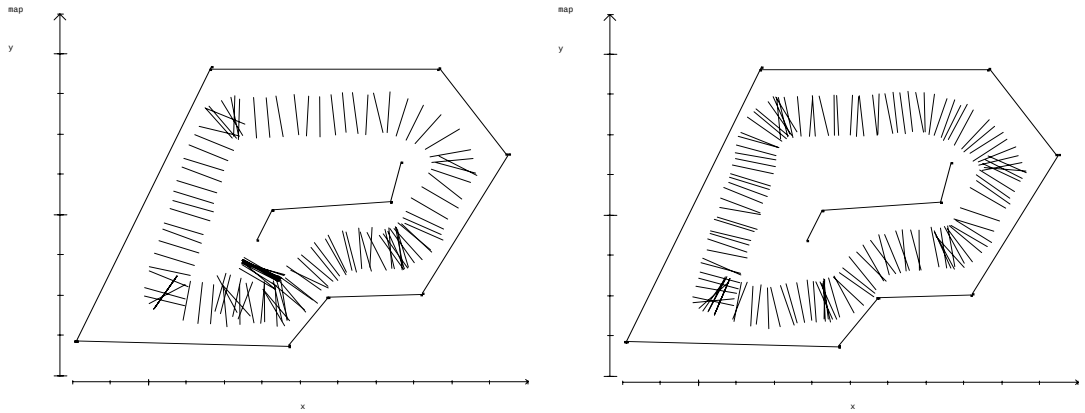


Figure 84: a. Trajectoire (sens contraire aux aiguilles d'une montre) du robot lors du premier tour de l'environnement. b. Trajectoire lors du deuxième tour de l'environnement. Chaque trait correspond à l'axe des roues et donne la position du Khepera toutes les 100 ms

5.5.4.1 La dynamique du réseau

Si le comportement visé, à savoir l'évitement d'obstacles, est identique à celui de la première expérience (section 5.3), le fonctionnement interne du réseau est lui totalement différent, car l'apprentissage et la plasticité synaptique introduisent un degré de liberté supplémentaire à la dynamique du réseau. Ce ne sont plus uniquement les activités de neurones qui varient et génèrent le comportement, mais aussi les valeurs synaptiques.

La figure 85 illustre, selon la même disposition des synapses de la figure 82b, la variation des poids synaptiques en fonction du temps et pendant les 30 premières secondes de fonctionnement du réseau. Sur chaque graphique, des indications permettent de connaître les caractéristiques génétiques de la synapses, déterminées par l'évolution et inscrites dans le génotype. Le premier symbole indique si la synapse est de type driving (d) ou modulatory (m). Le deuxième paramètre indique le gain d'apprentissage arrondi. La troisième indication donne le type d'apprentissage effectué sur la synapse: hebbien pur (hebb), postsynaptique (post), présynaptique (pre) ou synchrone (syn). La caractéristique inhibitrice ou excitatrice de la synapse est donnée par l'axe des ordonnées (positif si la synapse est excitatrice, négatif si la synapse est inhibitrice).

On peut généralement observer qu'un premier apprentissage rapide de quelques secondes permet d'atteindre un état dynamique stable, sur la base duquel se développe le comportement de suivi de mur, basé sur des variations cycliques de plus petite intensité et bien calibrées.

En analysant plus en détail les graphiques de la figure 85, nous pouvons mieux comprendre la dynamique du système et le rôle que jouent les différentes synapses. Afin de simplifier l'explication, nous allons regrouper les synapses et en analyser le rôle. Le différents groupes de synapses sont caractérisés par une lettre qui fait référence à la figure 85.

Tout d'abord nous pouvons observer que l'unité de la couche cachée n'est pas utilisée pour élaborer des données sensorielles pour l'évitement d'obstacles (groupe A). En effet, les connexions entre cette unité et les capteurs frontaux ont toutes des poids synaptiques nuls, dus aux taux d'apprentissage nuls, à l'exception du deuxième capteur de gauche. Le poids synaptique de ce dernier reste toutefois proche de zéro dans les conditions normales de suivi de mur. L'unité cachée est toutefois maintenue active par les connexions avec les capteurs arrière et par le rebouclement fait par la connexion modulatrice excitatrice (groupe B). Grâce aux fortes connexions excitatrices qui lient l'unité cachée aux deux neurones moteurs (groupe C), celle-ci active les moteurs et pousse ainsi le robot à avancer.

Un autre groupe de connexions presque nulles et à gain d'apprentissage aussi nul, est celui qui relie les capteurs de gauche avec les deux moteurs (groupe D). Ceci est dû au comportement de suivi de la paroi fait sur la droite. La seule synapse qui a un gain d'apprentissage non-nul, est la synapse qui relie le capteur tout à gauche, avec le moteur de gauche. Cette connexion excitatrice permet d'éviter des obstacles qui apparaissent sur la gauche, par exemple pendant les premières périodes d'approche d'une parois.

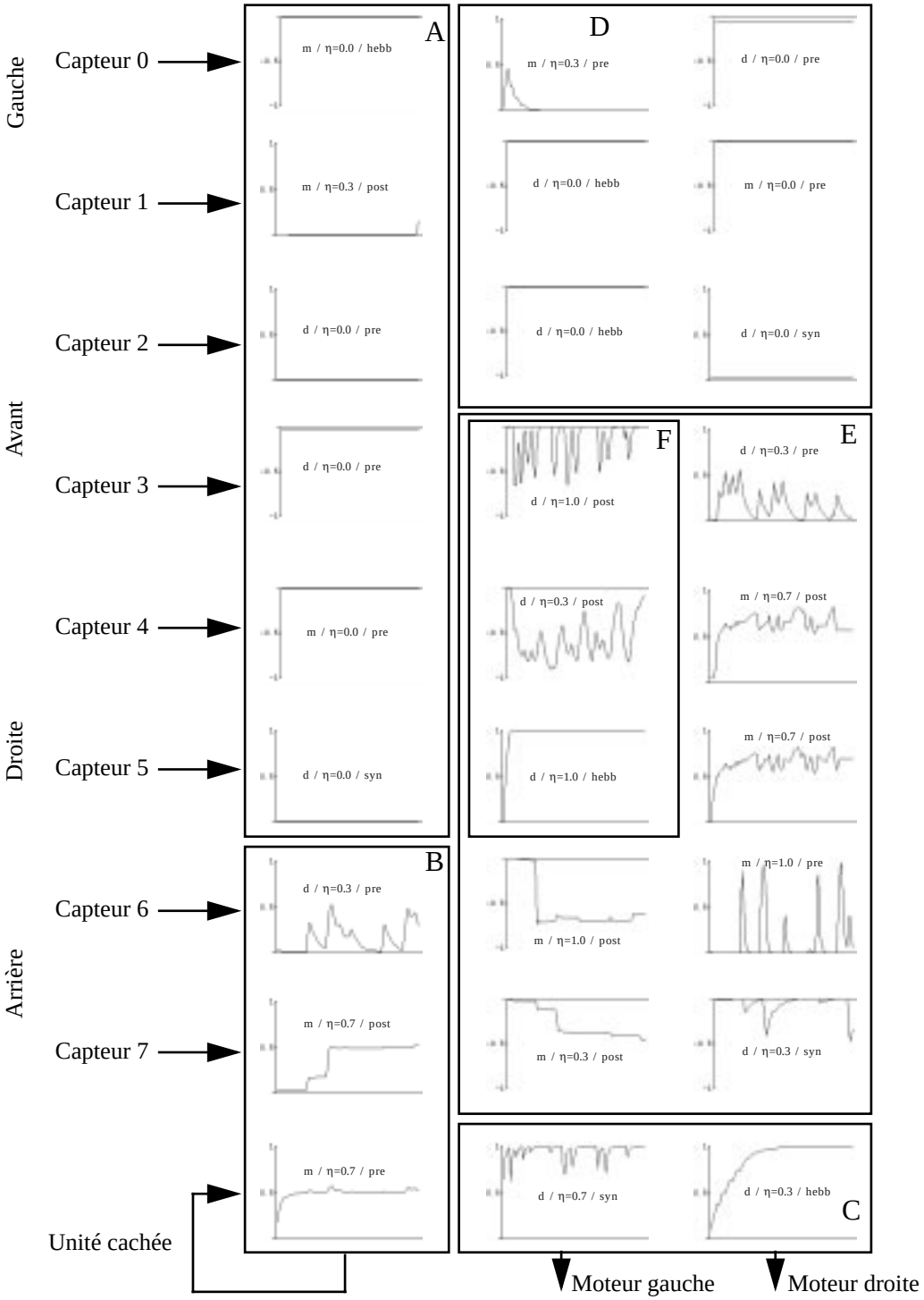


Figure 85: Poids synaptiques mesurés pendant les 30 premières secondes d'apprentissage du robot. Les caractéristiques génétiques de chaque synapse sont données au niveau de chaque graphique: Le premier symbole indique si la synapse est de type driving (d) ou modulatory (m). Le deuxième paramètre indique le gain d'apprentissage (arrondi), la troisième indication donne le type d'apprentissage effectué sur la synapse (hebb = pure hebb, post = postsynaptique, pre = présynaptique et syn = synchrone). La caractéristique inhibitrice ou excitatrice de la synapse est donnée par l'axe des ordonnées (positif = excitatrice, négatif = inhibitrice)

Le contrôle du comportement du robot se fait dans les connexions qui relient les capteurs de droite et arrière, avec les deux moteurs (groupe E). Les connexions qui portent au moteur de droite sont essentiellement excitatrices, sauf pour la connexion avec le capteur arrière gauche. Cette excitation, ajoutée à la connexion fortement excitatrice du neurone de la couche cachée, fait que la roue de droite va essentiellement seulement en avant. Cette poussée est modulée par les deux capteurs de droite à connexion modulatrice. Le capteur avant droite peut, de son côté, donner des fortes poussées avec sa connexion driving qui, associées aux encore plus fortes inhibitions de la connexion également driving sur le moteur gauche, permettent de reculer légèrement en faisant des virages rapides lors de la détection frontale d'obstacles. Nous pouvons observer que l'amplitude des poids de ces deux connexions diminue tout au long de la vie, ce qui correspond à un affinement du comportement de suivi.

Les synapses les plus importantes sont les trois qui relient les capteurs de droite avec le moteur de gauche (groupe F). La synapse qui correspond au capteur tout à droite, toujours un peu actif lors du suivi de parois, a un poids qui est saturé à un lors des premiers instants de vie du réseau. Cette connexion de type driving provoque une poussée continue sur la roue de gauche, ce qui tend à pousser le robot contre la parois. Les deux autres capteurs permettent de mesurer bien mieux la direction de la parois, et les connexions entre ces capteurs et le moteur de gauche permettent d'utiliser cette information pour compenser, en inhibant, la poussée de base du premier capteur. Ces compensations ont un caractère oscillatoire qui permet de vérifier en continuation la courbure de la parois.

Enfin nous pouvons remarquer que les différentes règles d'apprentissage ne sont pas utilisées dans la même proportion et pour les mêmes fonctionnalités. La règle hebbienne pure, en particulier, a été utilisée seulement sur cinq synapses, dont trois avec un gain d'apprentissage nul. Les deux autres synapses utilisent la règle hebbienne afin de saturer le poids synaptique à un, de façon plus ou moins rapide selon le gain d'apprentissage utilisé. Autrement les règles pre- et postsynaptiques sont nettement plus utilisées que la règle synchrone. Cette dernière n'est utilisé que dans les connexions entre la couche cachée et les neurones moteurs. Ceci montre que le rôle de l'unité caché est très lié à ceux des moteurs, avec une influence mutuelle importante.

5.5.4.2 Les comportements appris et les comportements héréditaires

Afin de mieux comprendre l'influence de l'aspect héréditaire et de l'apprentissage, le robot a été placé dans différentes situations.

Une première expérience a consisté à mettre le robot dans un espace ouvert, sans obstacles. Dans cet environnement, le robot a un comportement qui alterne des longues trajectoires rectilignes suivies de rotations rapides vers la gauche. Si ensuite le robot est remis dans l'environnement dans lequel il a subi l'évolution, il acquiert rapidement le comportement de suivi de mur. Remis dans l'espace ouvert, le robot a un tout autre comportement qu'au début, et alterne des rotation vers la droite avec des arrêts brusques. Ce dernier comportement est compréhensible dans l'optique de la recherche d'une parois, comportement qui n'était pas présent dans les mêmes conditions au début de la vie et qui a été développé lors de l'introduction du robot dans l'environnement d'évolution.

Une autre expérience similaire a consisté à placer le robot dans une environnement complètement fermé, comme le montre la figure 86. Aussi dans ce cas le robot a été confronté avec un environnement particulier qui limite la perception à des conditions fixes. Si dans l'expérience précédente les capteurs ne pouvaient rien percevoir car l'environnement était sans obstacles, ici les capteurs du robot sont toujours saturés, pour n'importe quelle action du robot. Si le robot est placé dans cet environnement au début de sa vie, il actionne ses roues pour avancer et pousse contre la parois (figure 86a). Si on place le robot dans cet environnement après l'avoir placé dans son environnement d'évolution, le robot tourne sur place, comportement qui pourrait être utile pour trouver une sortie (figure 86b). On dirait donc que la première période de vie, si vécue dans un environnement normal, permet au robot d'apprendre comment se comporter face aux parois, chose qu'il n'apprend pas s'il est laissé dans un environnement complètement fermé depuis sa naissance.

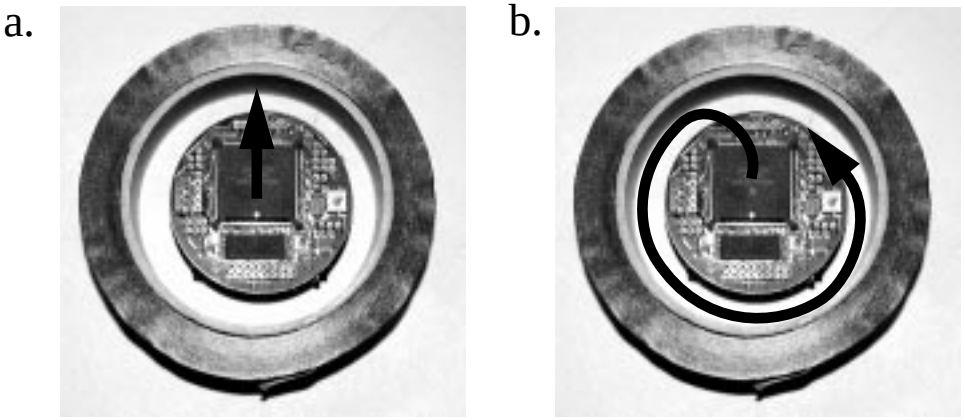


Figure 86: Robot placé dans un environnement fermé. Si le robot est placé dans cet environnement au début de sa vie, il développe le comportement décrit en a), à savoir il pousse contre la parois. Si le robot est placé dans cet environnement seulement après avoir été placé dans une environnement *normal*, il développe le comportement décrit dans b), à savoir il semble chercher une sortie en tournant sur lui-même

5.5.5 Discussion

Grossièrement cette expérience et celle de la section 5.3 montrent une efficacité similaire. Dans les deux cas le robot effectue un bon évitement d'obstacles et ne reste pas bloqué comme l'était le type de véhicule de Braitenberg pris comme référence. Aussi dans quelques choix des solutions il y a similarité: Dans les deux cas le robot va dans la direction qui lui permet de mieux détecter les obstacles et les vitesses d'avancement se ressemblent.

Prises plus en détail, ces deux expérience montrent toutefois des différences importantes et intéressantes. La première (section 5.3) a permis de développer un comportement uniquement instinctif, car hérité génétiquement, et fortement réactif. Dans la dernière expérience le comportement instinctif est très limité (ligne droites et rotations sur la droite) alors que le comportement d'évitement est appris lors de l'interaction avec l'environnement. Le comportement en lui-même est moins réactif, car il s'adapte à l'environnement à court et long terme. Dans le suivi de parois, par exemple, le robot

accélère si la parois se montre assez longue. Dans son apprentissage à plus long terme et au cours des explorations de l'environnement, le robot adapte son comportement pour obtenir une trajectoire plus lisse. Dans cette deuxième expérience il y a ainsi une bien plus grande adaptivité et richesse comportementale. L'interaction avec l'environnement prend une plus grande importance, car exploitée à chaque instant pour une meilleure adaptation à la situation particulière. A l'adaptation génétique, qui fait une évaluation moyenne sur toute la durée de la vie, s'ajoute l'apprentissage continu, réalisé à chaque instant. On arrive ainsi à exploiter l'interaction avec une parois longue ou courte, ainsi qu'à exploiter la forme de l'environnement dès le deuxième tour.

Dans cette expérience il a aussi été possible d'observer des similitudes et des différences avec le monde biologique. Il est clair que des comparaisons de ce type doivent être faites avec prudence, car nous sommes loin de simuler la complexité biologique, mais il est intéressant de les mettre en évidence pour voir quels mécanismes sont communs ou différents dans les mondes biologique, qui a servi d'inspirateur, et robotique, que l'on a obtenu avec la méthode évolutionniste. L'aspect similaire le plus intéressant est obtenu au niveau comportemental. L'expérience de manque sensoriel lors de la naissance de l'individu, qui ne lui permet pas de développer ses facultés, a aussi été observé dans le monde animal [Wiesel82]. A côté de cette similitude il y a une différence intéressante: Dans les modèles biologiques on a l'habitude de considérer les poids synaptiques comme relativement stables ou qui changent lentement. Dans le réseau développé lors de cette dernière expérience, certains changements de poids synaptiques sont rapides et jouent un rôle presque au même niveau que les signaux qu'il pondèrent. La présence d'un tel type de synapse n'est d'ailleurs pas exclu en nature. Il n'a pas été rencontré jusqu'à présent, mais la difficulté de la mesure de l'efficacité synaptique ne permet pas d'exclure cette hypothèse.

5.6 Discussion sur l'approche évolutionniste

Dans les trois expériences présentée dans ce chapitre, l'approche évolutionniste s'est montrée très bien adaptée au développement de systèmes de contrôle de robots mobiles autonomes et ceci pour plusieurs raisons:

- L'approche évolutionniste base son fonctionnement sur l'interaction entre robot et environnement. Cet aspect est non seulement vital pour le succès du développement du système de contrôle, mais est un point qui pose de gros problèmes aux approches classiques par sa complexité. L'approche évolutionniste arrive à éviter la modélisation de cette interaction tout en l'utilisant dans son processus.
- L'évolution du système de contrôle dans le corps du robot permet d'exploiter au mieux cette interaction et de créer un système cohérent. L'évolution du système de contrôle prend naturellement en compte les caractéristiques et les besoins physiques du robot jusqu'à aller à créer des comportements pour les satisfaire (comme par exemples dans le cas du besoin de recharger les batteries) sans que ceci soit défini dans la fonction d'évaluation.

- Le mécanisme d'évolution, s'il est bien mis en oeuvre et exploité, permet de donner au robot une autonomie de décision très élevée. Ceci permet de résoudre des gros problèmes tels que le choix des représentations internes, leur exploitation ou encore le choix des lois d'apprentissage, comme l'ont démontré la deuxième et la troisième expérience.
- Dans ces expériences nous avons appliqué les algorithmes génétiques à des réseaux de neurones. L'utilisation des algorithmes génétiques permet de façonner de façon très fine le système de contrôle, que ce soit un réseau de neurones, comme dans notre cas, ou d'autres systèmes, comme par exemple des *classifier systems* [Colombetti93].

Certains de ces aspects ont été mis en évidence aussi par d'autres groupes de recherche [Cliff94][Brooks94], mais peu de ces travaux considèrent l'implémentation sur un système réel (voir aussi [Husband94] pour une vue d'ensemble sur ce domaine). Or l'adaptation du système à un environnement simulé, donc maîtrisable et modélisable, n'a qu'un intérêt limité. Ce travail met en évidence qu'il est possible et intéressant de réaliser l'évolution sur un système réel, en attaquant le problème de l'interaction robot-environnement dans sa complexité réelle.

Les aspects très intéressants de l'approche présentée sont contrebalancés par un grand problème, à savoir la durée d'évolution. La deuxième expérience a bien montré la croissance importante du temps nécessaire au développement, en fonction de la complexité de la tâche. Cette croissance est difficilement quantifiable et prévisible pour des nouvelles expériences, car la complexité même de la tâche ne l'est pas. On voit toutefois clairement que l'utilisation de cette technique ne peut pas être étendue de façon directe à des applications réelles et industrielles. Il est donc nécessaire, si on veut profiter des mécanismes intéressants de l'approche évolutionniste, d'améliorer la méthode présentée pour la rendre utilisable. Ces modifications peuvent aller dans plusieurs directions:

- La granularité des éléments soumis à l'évolution (dans notre cas les poids synaptiques, les règles d'apprentissage et d'autres paramètres des neurones d'un réseau fixe) peut être plus grossière, les composants de base pouvant être, par exemple, des comportements élémentaires. Cette approche a le désavantage de limiter le champ d'action de l'algorithme génétique, en lui laissant un choix restreint.
- L'algorithme génétique peut être implémenté sur une réelle population de robots. Cette approche a le désavantage de nécessiter un équipement important. Elle a toutefois l'avantage de pouvoir tenir compte de l'aspect collectif de l'application.
- Il est possible, lors des premières générations, de laisser plus de place à la simulation, pour ensuite retourner à l'implémentation réelle.

Cette dernière possibilité, discutée en détail dans [Nolfi94], semble être la plus prometteuse. D'autres groupes ont utilisé cette approche: Colombetti et Dorigo [Colombetti93] font évoluer en simulation un système de contrôle basé sur des *classifier systems*, pour enfin le tester sur le robot réel. Dans cette direction les travaux les plus prometteurs restent ceux de Nolfi [Nolfi95] qui arrive à obtenir, par évolution, des comporte-

ment de recherche, reconnaissance, prise et transport d'objets, testés en phase finale sur un robot réel.

Une autre limitation des expériences présentées dans ce travail par rapport à des applications réelles, consiste dans le fait que l'on a travaillé avec un robot miniature. La miniaturisation a facilité énormément l'application de la méthode évolutionniste. La résistance mécanique du robot, augmentée par la miniaturisation, joue en effet un rôle essentiel dans l'application des algorithmes génétiques, essentiellement lors des premières générations. Grâce à cet effet, Khepera s'est affirmé comme étant l'outil idéal pour ce type d'approche et est actuellement utilisé par les plus importantes équipes de recherche en robotique évolutionniste. Le fait que ces chercheurs aient adopté Khepera ou d'autres équipements spécialement bien adaptés à l'évolution et qu'ils soient toujours loin des applications [Harvey94] prouve une fois de plus la difficulté du passage à l'implémentation sur des robots mobiles de taille réelle. Les applications industrielles impliquent des robots adaptée à l'environnement dans lequel ils agissent. En taille réelle, il est impensable d'appliquer la méthode évolutionniste comme nous l'avons fait ici. Pour pouvoir exploiter les aspects intéressants de l'approche évolutionniste, une solution envisageable serait, comme indiqué auparavant, de dégrossir le problème par simulation. Le passage entre simulation et application réelle pourrait être aidé par une étape intermédiaire, utilisant des robots mobile miniature comme nous l'avons fait dans ce travail. Cette combinaison de simulation - essais miniaturisé - application réelle est une des possibles continuations de ce travail. Le robot Koala, présenté brièvement dans la section 2.3, a été conçu dans cette optique. Avec sa structure très similaire à Khepera, il pourrait permettre la continuation de l'évolution vers des applications réelles.

Si des travaux sur l'évolution du système de contrôle doivent continuer à être fait sur des systèmes réels, il existe un large nombre de travaux de simulation sur la morphologie même des agents qui peuvent très mal être traduits en pratique. Mis a part pour certaines techniques très particulière, comme l'utilisation de tubes d'installations électriques par [Jansen94], il serait intéressant de disposer de matériel réellement évolutif, du point de vue électronique et matériel.

6 DISCUSSION ET CONCLUSIONS

Toutes les expériences présentées dans ce travail ont un point en commun: Elles mettent en évidence le rôle clé de l'interaction robot-environnement. C'est à ce niveau que se situe la complexité que beaucoup de méthodes de conception ne sont pas arrivées à surmonter. Une bonne connaissance de cette interaction et son exploitation judicieuse a permis déjà des résultats intéressants, comme ceux de Brooks et d'autres groupes qui se sont penchés sur des problèmes précis. Dans ce travail, cet aspect a été illustré par les deux premières expériences, l'une collective et l'autre coopérative. La prise en compte et l'utilisation, lors de la conception, de l'interaction entre robot et environnement permet d'effectuer des tâches plus complexes que ce que l'on peut attendre simplement du programme de contrôle. Malheureusement on ne peut atteindre une si bonne connaissance de l'interaction robot-environnement que pour des cas très limités et très simples.

Pour surmonter ce problème on peut permettre au robot de saisir de façon autonome les caractéristiques de cette interaction robot-environnement. De cette façon on peut obtenir des systèmes adaptatifs comme celui de la troisième expérience, avec de l'apprentissage. Les mécanismes d'adaptation sont toutefois souvent très dédiés au problème qu'ils doivent résoudre, ce qui les rend presque aussi limités que les systèmes conçus manuellement et décrits au début.

Il faudrait donc revoir les méthodes de conception et les baser sur cette interaction, dans la direction illustrée par la quatrième expérience. C'est l'interaction qui devrait dicter directement et très librement les choix de base du développement du système, comme celui des représentations internes, illustrés dans la cinquième expérience. Afin d'exploiter le plus possible cette interaction, au niveau de la méthode de conception comme au niveau d'autres méthodes d'adaptation, l'apprentissage peut être introduit comme le montre la sixième expérience. Les résultats prouvent que l'interaction robot-environnement n'est pas seulement un aspect incontournable de la robotique mobile autonome, mais peut être une base de développement.

Les expériences faites ici et basées sur l'approche évolutionniste essaient d'illustrer une nouvelle voie de conception, mais restent très limitées à cause des problèmes classiques. L'apport du concepteur reste très important, et canalise souvent de façon très stricte les résultats. Le choix du nombre de neurones, de la taille du réseau, du type de règles d'apprentissage, du type de capteurs, de la façon dont les roues sont contrôlées, etc., restent un facteur décisif pour la réussite de la conception. On reste ainsi loin d'une conception facilitée au point de ne pas demander au concepteur d'avoir des connaissances du problème.

Un autre aspect important d'une méthode de conception est qu'elle devrait permettre au concepteur d'arriver au résultat avec un effort moins important que si le développement était fait complètement manuellement. On critique ainsi souvent les méthodes évolutionnistes en affirmant que le temps de conception de la fonction d'évaluation correspond à celui de résolution du problème par d'autres méthodes. Dans le cas de l'algorithme d'évitement d'obstacles, ceci est sûrement vrai. Dans le cas de la navigation vers

le nid je pense qu'on s'approche à des performances similaires ou supérieures à celles d'autres méthodes de conception. Je trouve en effet très difficile de concevoir un réseau de neurones, ou une autre structure de contrôle de complexité similaire, de taille si réduite et toutefois capable des performances obtenues par la méthode évolutionniste, en particulier la rentrée régulière dans les 20 secondes. Ce résultat ne donne pas un avantage net à la méthode évolutionniste, mais est encourageant et prouve que l'approche mérite d'être approfondie.

Enfin les expériences évolutionnistes on mis en évidence un gros désavantage de cette approche, à savoir la durée d'évolution. C'est surtout à ce niveau qu'il a un obstacle important pour le développement d'applications réelles. C'est donc à ce niveau que doivent se concentrer les efforts de recherche si on désire utiliser cette technique comme une vraie méthodologie de développement de robots mobiles autonomes. Plusieurs voies sont possibles et ont été évoqués dans la section 5.6. La plus prometteuse semble être celle de la combinaison réel-simulé développée par étapes de complexité. Si des résultats ont déjà été obtenus [Nolfi95][Jakobi95][Michel96], la complexité des comportements développés a toujours été limitée. Cette méthode doit donc être mise à l'épreuve avec des applications réelles.

Malgré l'impression que la simulation peut prendre une place dans les méthodes de développement, l'importance de la relation entre le robot et son environnement est trop grande et l'interaction trop complexe pour qu'en la modélisant et simulant on puisse espérer avoir des résultats d'un niveau suffisant pour réaliser des applications. L'utilisation ou du moins la validation sur des robots réels est donc essentielle dans ces expériences.

Le gros problème de l'autonomie, même s'il mérite une place à part, est aussi intimement lié à l'interaction robot-environnement. L'autonomie n'est en effet que la capacité de gérer cette interaction. Créer l'autonomie sans tenir compte de cette interaction est donc absurde. Il est ainsi fondamental de créer l'autonomie sur la base de cette interaction. La cinquième expérience montre bien qu'il est possible de donner au système l'autonomie en ce qui concerne le choix des structures de contrôle. Si ce choix est fait sur la base des besoins du robot dans la gestion de son interaction avec l'environnement, ceci peut porter à des résultats très intéressants, comme la gestion de la navigation faite avec un réseau de neurones ayant une couche cachée composée de 5 neurones seulement.

Ce dernier exemple ainsi que la toute première expérience montrent aussi qu'une bonne exploitation de l'interaction robot-environnement peut porter à des simplifications considérables dans la structure de contrôle du robot. Cet aspect a beaucoup de choses en commun avec le côté coopératif vu dans la première expérience, ainsi que la "coopération" des neurones à l'intérieur d'un réseau. La somme de comportements élémentaires (travail collectif), ou de fonctions élémentaires (réseaux de neurones) pose déjà beaucoup de problèmes de compréhension, même pour des systèmes relativement simples. Au niveau de la somme des différents aspects de l'interaction robot-environnement on a encore plus de problèmes de compréhension, car à l'aspect de combinaison s'ajoute la complexité de chaque élément. Il est donc essentiel de continuer à étudier des méthodes qui tiennent compte de ces aspects.

L'interaction robot-environnement est probablement non seulement une des clés d'accès aux problèmes de la complexité dans la direction du développement d'applications, mais aussi une clé d'accès à l'autonomie.

En guise de conclusion, on peut noter que ce travail, tout en essayant de poursuivre des lignes prometteuses qui avaient déjà fait l'objet d'études précédentes, contient quelques contributions originales pour le domaine de la robotique mobile:

- L'idée, aujourd'hui bien acceptée, qu'il est nécessaire de travailler avec un robot mobile a été traduite en une méthodologie et des outils de travail stables et performants. L'électronique développée par Edo Franzi et la mécanique conçue par André Guignard ont été utilisées pour créer un environnement de travail convivial, efficace et robuste. Ces outils et cette méthode de travail ont contribué à répandre l'utilisation de robots mobiles réels, et sont utilisés dans leur version commerciale par environ 350 utilisateurs.
- Les trois dernières expériences ont permis de mettre en évidence la faisabilité ainsi que les avantages et les inconvénients de l'utilisation d'algorithmes génétiques sur des robots réels, ce qui n'avait jamais été fait auparavant. La première de ces trois expériences est reconnue comme étant le premier essais en absolu de cette approche. Les deux autres expériences ont mis en évidence des éléments très intéressants pour la compréhension des caractéristiques de la méthode évolutionniste. Ici aussi, l'outil Khepera est pratiquement le seul utilisé dans la communauté scientifique pour ce genre d'expérience, ce qui prouve la validité des outils mis en place.

7 REMERCIEMENTS

Ce travail de recherche a été réalisé au Laboratoire de Microinformatique de l'Ecole Polytechnique Fédérale de Lausanne. Il a démarré grâce à l'esprit innovateur du Prof. J.-D. Nicoud, que je remercie de m'avoir lancé dans ce domaine et soutenu dans ma démarche.

Ce travail n'aurait toutefois jamais vu le jour sans la collaboration des deux experts en électronique et mécanique qui ont conçu et réalisé le robot Khepera: Edo Franzi et André Guignard. Leur support personnel et professionnel a été fondamental pour ma motivation et pour la réussite de tout le projet Khepera. Je les remercie donc de tout coeur en espérant travailler avec eux dans la suite de ce projet.

Je dois une bonne partie de l'inspiration “théorique” de ce travail à trois personnes: Paul Verschure, Philippe Gaussier et Dario Floreano. Paul Verschure m'a aidé à entrer dans le domaine des réseaux de neurones et de l'apprentissage, en me mettant à disposition son savoir faire dans le domaine. Philippe Gaussier a initié les expériences collectives au LAMI, et a été le précurseur des expériences que j'ai présenté dans ce travail. Dario Floreano est la personne de laquelle j'ai le plus appris sur la robotique autonome et sur comment l'approcher, soit dans la synthèse que dans l'analyse. Je le remercie donc tout particulièrement pour son apport tant professionnel que personnel.

Autour de ces personnes, tout l'environnement du LAMI a joué un rôle important dans le déroulement de mon travail, et je remercie tous ceux qui l'ont rendu agréable, productif et professionnel. Au dehors du LAMI je remercie Beatrice Kälin, qui a eu la patience et le courage de relire un bonne partie de ce travail.

Je remercie enfin Judith, ma femme, qui a su me motiver, me donner son temps et sa patience pour que je puisse terminer ce travail...

8 RÉFÉRENCES

[Almassy92] N. Almassy and P. Verschure, “Optimizing self-organizing control architectures with genetic algorithms: the interaction between natural selection and ontogenesis”, in Männer and Manderick Eds., Parallel Problem Solving from Nature, 2: proceedings of the Second Conference on Parallel Problem Solving from Nature, Brussels, Belgium, Amsterdam, Elsevier, 1992, pp. 451-460.

[Beckers94] R. Beckers, O.E. Holland and J.L. Deneubourg, “From Local Actions to Global Tasks: Stigmergy and Collective Robotics”, in Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, R. Brooks and P. Maes, Eds., Boston, MA, MIT Press, 1994.

[Bejczy94] A. Bejczy, G. Bekey, R. Taylor, and S. Rovetta, "A research methodology for tele-surgery with time delays", First Int. Symposium on Medical Robotics and Computer Assisted Surgery, Sept 1994.

[Bessière92] P. Bessière, “Genetic algorithms applied to formal neural networks: parallel genetic implementation of a Boltzmann machine and associated robotic experimentation”, in Toward a Practice of Autonomous Systems. Proceedings of the First European Conference on Artificial Life, F.J. Varela and P. Bourgine, Eds., MIT Press, Cambridge, MA, 1992, pp. 310-314.

[Brutzman95] D. Brutzman, “Virtual World Visualization for an Autonomous Underwater Vehicle”, Proceedings of the IEEE Oceanic Engineering Society Conference OCEANS 95, San Diego California, October 12-15 1995, pp. 1592-1600.

[Braitenberg84] V., Braitenberg, “Vehicles. Experiments in Synthetic Psychology”, MIT Press, Cambridge, 1984.

[Brooks83] R.A. Brooks, “Solving the Find-Path Problem by a Good Representation of Free Space”, IEEE Trans. on Systems, Man and Cybernetics, SMC-13 No.3, March 1983, pp. 190-197.

[Brooks86] R. A., Brooks, "A robust layered control system for a mobile robot", IEEE Robotics and Automation, RA-2, 1986, pp. 14-23.

[Brooks90] R.A. Brooks, “Elephants Don’t Play Chess”, Robotics and Autonomous Systems 6 (1990), pp. 3-15.

[Brooks91] R.A. Brooks, “Intelligence without representation”, Artificial Intelligence, 47, 1991, pp. 139-159.

[Brooks91b] R.A. Brooks, “Intelligence without Reason”, MIT AI Lab Memo #1293, April 1991.

[Brooks93] R.A. Brooks, and A. S. Lynn, “Building Brains for Bodies”, MIT AI Lab Memo #1439, August 1993.

[Brooks94] R.A. Brooks, and P. Maes, editors, “Proceedings of the Fourth Workshop on Artificial Life”, MIT Press, Boston, MA, 1994.

[Brooks96] R.A. Brooks, Copy of the slides of the presentation held at the Evolutionary Robotics ‘96 Conference, April 1995, Tokyo, Japan, 1996.

[Chatila94] R. Chatila, “Control Architectures for Autonomous Mobile Robots”, in Proceedings of the

conference From Perception to Action, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 254-265.

[Cheneval95] Y. Cheneval, “Packlib, an Interactive Environment to Develop Modular Software for Data Processing”, in J. Mira and F. Sandoval Eds, From natural to Artificial Neural Computation, Springer, berlin Heidelberg, 1995, pp. 673-682.

[Cliff94] D. Cliff, P. Husbands, J. Meyer and S. W. Wilson, editors, “From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour”, MIT Press-Bradford Books, Cambridge, MA, 1994.

[Colombetti93] M. Colombetti, and M. Dorigo, “Learning to Control an Autonomous Robot by Distributed Genetic Algorithms”, In in From Animals to Animats II: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, J. Meyer, H. L. Roitblat and S.W. Wilson, Eds., MIT Press-Bradford Books, Cambridge, MA, 1993.

[Conquet94] F. Conquet, Z. I. Bashir, D. C. H. Daniel, F. Ferraguti, F. Bordi, K. Franz-Bacon, A. Reggiani, V. Matarese, F. Conde, G. L. Collingridge and F. Crepel, “Motor deficit and impairment of synaptic plasticity in mice lacking mGluR1”, Nature, vol. 372, 1994, pp. 237-243.

[Crick89] F. Crick, “The Recent Excitement About Neural Networks”, Nature, vol. 337, 1989, pp. 129-132.

[Dario92] P. Dario, R. Valleggi, M.C. Carrozza, M.C. Montesi and M. Cocco, “Microactuators for microrobots: A critical survey”, Journal of Micromechanics and Microengineering, 2, 1992, pp. 141-157.

[Deneubourg83] J.L. Deneubourg, J.M. Pasteels and J.C. Verhaeghe, “Probabilistic Behavior in Ants: A Strategy of Errors”, Journal of Theoretical Biology, vol. 105, 1983, pp. 259-271.

[Deneubourg91] J.L. Deneubourg, S. Goss, N. Franks, A. Sendova, A. Franks, C. Detrin, and L. Chatier, “The dynamics of collective sorting: Robot-like ant and ant-like robot”, In Simulation of Adaptive Behavior: From Animals to Animats, J.A. Mayer and S.W. Wilson editors, MIT Press, 1991, pp. 356-365.

[Dennett91] J. Dennett and T. Smithers, “Lego Vehicles: A Technology for Studying Intelligent Systems”, In Simulation of Adaptive Behavior: From Animals to Animats, J.A. Mayer and S.W. Wilson editors, MIT Press, 1991, pp. 540-549.

[Edelman89] G.M. Edelman, G.N. Reeke, W. Gali, G. Tononi, D. Williams and O. Sporns, “Synthetic neural modelling applied to a real-world artifact”, Proc. Natl. Acad. Sci.USA, 1989, pp. 7267-7271.

[Engel92] A.K. Engel, P. Koenig, A.K. Kreiter, T.B. Schillen and W. Singer, “Temporal coding in the visual cortex: new vistas on integration in the nervous system”, Trends in Neuroscience, vol. 15, 1992, pp. 218-226.

[Fleury93] S. Fleury, T. Baron and M. Herrb, “Monocular Localisation of a Mobile Robot”, Proceedings of 3rd International Conference on Intelligent Autonomous Systems (IAS-3), Pittsburgh, Pennsylvania, February 15-18, 1993.

[Floreano93] D. Floreano, "ROBOGEN: A Software Package for Evolutionary Control Systems. Release 1.1.", Technical Report No. 93-01, Cognitive Technology Laboratory, AREA Science Park, Trieste, Italy, 1993.

[Franceschini72] N. Franceschini, “Pupil and pseudopupil in the compound eye of Drosophila”, In Information processing in the visual system of arthropods, ed. R. Wehner, Berlin: Springer, 1972, pp. 75-82.

[Franceschini91] N. Franceschini, J.-M. Pichon and C. Blanes, “Real time visumotor control: from flies to robots”, IEEE Fifth International Conference on Advanced Robotics, PISA (Italy), June 1991, pp. 91-95.

[Franceschini92] N. Franceschini, J.M. Pichon and C. Blanes, “From insect vision to robot vision”, In Phil. Trans. R. Soc. Lond. B, 337, 1992, pp. 283-294.

[Gat91] E. Gat, “ALFA: A Language for Programming Reactive Robotic Control Systems”, Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, Vol. 2, 1991, pp. 1116-1121.

[Gaussier94] Ph. Gaussier and S. Zrehen, “A Constructivist Approach fro Autonomous Agents”, Wiley & Sons, Ed. N. Thalman, 1994.

[Gaussier94b] Ph. Gaussier and S. Zrehen, “A topological Neural Map for On-Line Learning: Emergence of Obstacle Avoidance”, From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, D. Cliff, P. Husbands, J. Meyer and S. W. Wilson, editors, MIT Press-Bradford Books, Cambridge, MA, 1994, pp. 182-190.

[Gaussier94c] Ph. Gaussier and S. Zrehen, “Navigating with an Animal-Like Brain: A Neural Network for Landmark Identification and Navigation”, Proceedings of Intelligent Vehicles, Paris, Oct. 1994, MIT Press, 1994.

[Goldberg89] D.E. Goldberg, “Genetic algorithms in search, optimisation and machine learning”, Reading, MA, Addison-Wesley, 1989.

[Harnad90] S. Harnad, “The symbol grounding problem”, Psysica D 42, 1990, pp. 335-346.

[Harvey94] I. Harvey, P. Husbands and D. Cliff, “Seeing The Light: Artificial Evolution, Real Vision”, in From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, D. Cliff, P. Husbands, J. Meyer and S.W. Wilson, Eds., MIT Press-Bradford Books, Cambridge, MA, 1994.

[Hebb49] D. O. Hebb, “The organisation of behavior”, New York, Wiley, 1949.

[Hérault92] J. Hérault, “Traitement du signal pour les réseaux de neurones naturels et artificiels”, Notes du XIIIe cours postgrade en informatique technique “Réseaux de neurones biologiques et artificiels”, Ecole Polytechnique Fédérale de Lausannne, avril 1992.

[Holland75] J.H. Holland, “Adaptation in natural and artificial systems”, The University of Michigan Press, Ann Arbor, 1975.

[Hopfield82] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilitie”, Proceedings of the National Academy of Sciences USA, vol. 79, 1982, pp. 2554-2558.

[Hoppen92] P. Hoppen, “Autonome Mobile Roboter”, Reihe Informatik, Band 87, Herausgegeben von K. H. Böhling, U. Kulisch, H. Maurer, Wissenshaftsverlag, Mannheim-Leipzig-Wien-Zürich, 1992.

[Husband94] P. Husbands, I. Harvey, D. Cliff and G. Miller, “The Use of Genetic Algorithms for the

Development of Sensorimotor Control Systems", in Proceedings of the conference From Perception to Action, IEEE Computer Society Press, Los Alamitos, CA, 1994.

[Jakobi94] N. Jakobi, "Evolving sensorimotor control architectures in simulation for a real robot", Master's thesis, Universit of Sussex, School of Cognitive and Computing Science. Unpublished.

[Jakobi95] N. Jakobi, P Husbands and I. Harvey, "Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics", in F. Morán, A. Moreno, J.J. Merelo and P. Chacón, eds, Advances in Artificial Life: Proceedings of the Third International Conference on Artificial Life, Springer-Verlag, 1995, pp. 704-720.

[Jansen94] T. Jansen, "Artifauna", in Additional documentation of the Proceedings of the conference From Perception to Action, LAMI-EPFL, Lausanne, 1994.

[Jones93] J.L. Jones and A.M. Flynn, "Mobile Robots. Inspiration to implementation", A.K. Peters, Wellesley, CA, 1993.

[Jutten88] C. Jutten, and J. Hérault, "Une solution neuromimétique au problème de séparation de sources", Traitement du Signal, Volume 5, No. 6, 1988, pp. 389-403.

[Kaelbling86] L.P. Kaelbling, "REX Programmer's Manual", SRI International Technical Note 381, May, 1986.

[Kahn91] P. Kahn, "Specification & Control of Behavioral Robot Programs", Proceedings SPIE Conference on Sensor Fusion IV,Boston, MA, Nov. 1991.

[Kay94] J. Kay and W.A. Phillips, "Activation Functions, Computational Goals and Learning Rules for Local Processing with Contextual Guidance", Technical Report CCCN-15, Centre for Cognitive and Computational Neuroscience, University of Stirling, April 1994.

[Knieriemen91] T. Knieriemen, "Autonome Mobile Roboter", Reihe Informatik, Band 80, Herausgegeben von K. H. Böhling, U. Kulisch, H. Maurer, Wissenshaftsverlag, Mannheim-Leipzig-Wien-Zürich, 1991.

[Kohonen88] T. Kohonen, "Self organization and associative memoryy", Second edition, Springer-Verlag, Berlin 1988.

[LabVIEW96] LabVIEW User Manual, Part Number 320999A-01, National Instruments Corporation, Austin TX, January 1996.

[Langton88] C. Langton, "Artificial Life", in Artificial Life, Santa Fe Inst. Studies in the Science of Complexity, C. Langton, Ed. Reading, MA: Addison-Wesley, 1988, pp. 1-47.

[Lüth94] T.C. Lüth and U. Rembold, "Extensive Manipulation Capabilities and Reliable Behaviour at Autonomous Robot Assembly", In Proc. of IEEE Int. Conf. on Robotics and Automation, San Diego, CA, 1994.

[Mead89] C. Mead, "Adaptive retina", in Analog VLSI implementation of neural systems, C. Mead and M. Ismail, Eds., chapter 10, Kluwer Academic Publishers, Boston, 1989, pp. 239-246.

[McCulloch43] W. S. McCulloch and W. Pitts, " A logical calculus of the idea immanent in nervous systems", Bulletin of mathematical Biophysics, No. 5, 1943, pp. 115-133.

[Michel96] O. Michel, "An Artificial Life Approach for the Synthesis of Autonomous Agents", Selected Papers of the AE95 Conference, J.-M. Alliot, E. Lutton, E. Ronald, M. Shoenauer and D. Snyers Eds., Springer, 1996.

[Minsky75] M. Minsky, “A framework for representing knowledge”, in P. Winston Ed., The psychology of computer vision, New York, Mc Graw-Hill, 1975, pp. 211-277.

[Mondada93] F. Mondada, E. Franzi, and P. Ienne, "Mobile robot miniaturization: A tool for investigation in control algorithms", in Proceedings of the Third International Symposium on Experimental Robotics 1993, Kyoto, Japan, T. Yoshikawa and F. Miyazaki, Eds., Springer Verlag, 1994.

[Montana89] D. Montana, L. Davis, “Training feed forward neural networks using genetic algorithms”, in Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, San Mateo, CA, 1989, Morgan Kaufmann.

[Newell72] A. Newell and H.A. Simon, “Human Problem Solving”, Englewood, Prentice Hall, 1972.

[Nilsson84] N.J. Nilsson, “Shakey the Robot”, Technical Note 323, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1984.

[Nolfi90] S. Nolfi, J. L. Elman and D. Parisi , “Learning and Evolution in Neural Networks”, Technical Report CRL90-19, University of California at San Diego, July 1990.

[Nolfi94] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics", in Proceedings of the Fourth Workshop on Artificial Life, R. Brooks and P. Maes, Eds., MIT Press, Boston, MA, 1994.

[Nolfi95] S. Nolfi and D. Parisi, “Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects”, In: M. Gori and G. Soda (eds), Topics in Artificial Intelligence, Proceedings of Fourth Congress of the Italian Association for Artificial Intelligence, Berlin, Springer Verlag, 1995.

[Pellerin95] C. Pellerin, “The service robot market”, Service Robot Journal, Vol. 1, Number 3, MCB University Press, 1995, pp. 17-20.

[Pomerleau93] A.D. Pomerleau, “Neural Network Perception for Mobile Robot Guidance”, Kluwer Academic Publishers, Boston-Drodrecht-London, 1993.

[Pylyshyn87] Z. W. Pylyshyn, ed., “The Robot’s Dilemma, The Frame Problem in Artificial Intelligence”, Ablex, Norwood, NJ, 1987, 2nd printing 1988.

[Rechenberg73] I. Rechenberg, “Evolutionstrategie: Optimierung technische Systeme nach Prinzipien der biologischen Evolution”, Friedrich Frommann Verlag, Stuttgart, 1973.

[Rumelhart86] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning Representations by Back-Propagation of Errors”, Nature, vol. 323, 1986, pp. 533-536.

[Schilling95] K. Schilling and Chr. Jungius, “Mobile Robots for Planetary Exploration”, in . Halme / K. Koskinen (eds.), Intelligent Autonomous Vehicles, 1995, pp. 110 - 120.

[Schofield95] M. Schofield, “Cleaning robots”, Service Robot Journal, Vol. 1, Number 3, 1995, MCB University Press, pp 11-16.

[Singer90] W. Singer, “Search for coherence: a basic principle of cortical self-organisation”, Concepts in Neuroscience, vol. 1, 1990, pp. 1-26.

[Smithers94] T. Smithers, “On Why Better Robots Make it Harder”, From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, D. Cliff, P. Husbands, J. Meyer and S. W. Wilson, editors, MIT Press-Bradford Books, Cambridge,

MA, 1994, pp. 64-72.

[Steels92] L. Steels, “The PDL reference manual”, VUB AI Lab memo. 92-5.

[Steels94] L. Steels, “A Case Study in the Behavior-Oriented Design of Autonomous Agents”, From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, D. Cliff, P. Husbands, J. Meyer and S. W. Wilson, editors, MIT Press-Bradford Books, Cambridge, MA, 1994, pp. 445-452.

[Steels95] L. Steels, “When are robots intelligent autonomous agents?”, Robotics and Autonomous Systems, Vol 15, Numbers1-2, July 1995, Elsevier, pp 3-10.

[Steels95b] L. Steels (ed.), The Biology and Technology of Intelligent Autonomous Agents, Springer, Heidelberg, 1995.

[Stewart94] J. Stewart, “The Implications for Understanding High-level Cognition of a Grounding in Elementary Adaptive Systems”, in Proceedings of the conference From Perception to Action, IEEE Computer Society Press, Los Alamitos, CA, 1994.

[Suchman87] L. A. Suchman, “Plans and Situated Actions”, Cambridge University Press, Cambridge, 1987.

[Touzet96] C. Touzet, "Neural reinforcement learning for behavior synthesis," CESA'96 (Computational Engineering in Systems Applications), IMACS Multiconference, Lille, France, 9-12 July 1996.

[Ulrich96] I. Ulrich, F. Mondada and J.-D. Nicoud, “Autonomous Vacuum Cleaner”, Robotics and Autonomous Systems, In Press.

[Verschure91] P. F. M. J. Verschure, and A. C. C. Coolen, “Adaptive Fields: Distributed representation of classically conditioned associations”, Network, 2, 1991, pp. 189-206.

[Verschure92] P.F.M.J. Verschure, B.J.A. Kröse and R. Pfeifer, "Distributed adaptive control: The self-organization of structured behavior", Robotics and Autonomous Agents, 9, 1992, pp. 181-196.

[Verschure92b] P.F.M.J. Verschure, “Modeling adaptive behavior: the dynamics of system environment interaction”, PhD thesis, University of Zurich, Switzerland, 1992.

[Widrow93] B. Widrow and M.A. Lehr, “Adaptive neural networks and their applications”, Int. J. Intelligent Systems, 8, 1993, pp. 453-507.

[Wiesel82] T.N. Wiesel, “Postnatal development of the visual cortex and the influence of the environment”, Nature, 299, pp. 583-591.

[Wolfram92] S. Wolfram, “Mathematica Reference Guide”, Addison-Wesley, 1993.

[Yuta91] S. Yuta, S. Suzuki and S. Ilda, “Implementation of a Small Size Experimental Self-Contained Autonomous Robot - Sensors, Vehicle Control, and Description of Sensor based Behaviour”, Proceedings of 2nd International Symposium on Experimental Robotics, Toulouse, France, June 25-27, 1991.

Curriculum Vitae

Francesco Mondada

né le 17.3.1967 à Locarno, TI

de nationalité suisse

Formation

1991	Diplôme d'ingénieur en microtechnique de l'EPFL
1991	Certificat de cours postgrade en visualisation scientifique et simulation graphique
1992	Certificat de cours postgrade en réseaux de neurones biologiques et artificiels
1993	Certificat de cours postgrade en nouvelles approches des bases de données
1993	Certificat de maîtrise en informatique

Expérience professionnelle

1991 -	Assistant de recherche et d'enseignement au Laboratoire de Microinformatique, EPFL
1995 -	Président du conseil d'administration et directeur des ventes de K-Team SA

Prix

1991	Prix PORTESCAP "d'excellence en microtechnique" pour le projet de diplôme "robot jongleur"
1993	Prix "Technologiestandort Schweiz" avec l'équipe du LAMI pour le projet "Khepera, un outil d'expérimentation en robotique mobile", élu parmi les meilleurs projets technologiques suisses et invité à la foire de Hannover

Publications

A. Basso, F. Mondada, C. Castelfranchi, "Reactive Goal Activation in Intelligent Autonomous Agent Architectures", AAI93 symposium on Abstract Intelligent Agents, Rome, January 1993.

F. Mondada, P. Verschure, "Modeling system-environment interaction: the complimentary roles of simulation and real world artifacts", ECAL'93, Brussels, March 1993, pp. 808-817.

F. Mondada, E. Franzi, P. Ienne, "Mobile robot miniaturization: a tool for investigation in control algorithms", ISER'93, Kyoto, October 1993.

S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics", in Proceedings of the Fourth Workshop on Artificial Life, R. Brooks and P. Maes, Eds., MIT Press, Boston, MA, 1994.

D. Floreano and F. Mondada, "Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot", in From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, D. Cliff, P. Husbands, J. Meyer and S.W. Wilson, Eds., MIT Press-Bradford Books, Cambridge, MA, 1994.

D. Floreano and F. Mondada, "Active Perception, Navigation, Homing, and Grasping: An Autonomous Perspective", in Proceedings of the PerAc conference, Ph. Gaussier and J.-D. Nicoud, Eds., Lausanne, Switzerland, IEEE Computer Society Press, 1994.

A. Martinoli and F. Mondada, "Collective and Cooperative Behaviours: Biologically Inspired Experiments in Robotics", Preprints of the 4th International Symposium on Experimental Robotics, ISER'95, Stanford, CA, June 30-July 2, 1995, pp. 2-7.

F. Mondada and D. Floreano, "Evolution of neural control structures: some experiments on mobile robotics", Robotics and Autonomous Systems, Vol. 16, Nr 2-4, December 1995, pp. 183-195.

D. Floreano and F. Mondada, "Evolution of Homing Navigation in a Real Mobile Robot", IEEE Transactions on Systems, Man and Cybernetics, Vol 26, Number 3, June 1996, pp. 396-407.

F. Mondada and D. Floreano, "Evolution and Mobile Robotics", In "Towards Evolvable Hardware", E. Sanchez and M. Tommasini Eds., Springer Verlag, Berlin, 1996, pp. 221-249.

I. Ulrich, F. Mondada and J.-D. Nicoud, "Autonomous Vacuum Cleaner", Robotics and Autonomous Systems, In Press.