Approche neuromimétique modulaire pour la commande d'un système robot-vision

Gilles Hermann

Année 2004 N° d'ordre : **04 MULH 0780/23**

Université de Haute-Alsace
U.F.R. des Sciences et Techniques

Thèse

Présentée pour obtenir le grade de

Docteur de l'université de Haute-Alsace

Discipline E.E.A.

par

Gilles HERMANN

Approche neuromimétique modulaire pour la commande d'un système robot–vision

---

(Arrêté Ministériel du 30 mars 1992)
Soutenue publiquement le 3 décembre 2004 devant le jury :

Président	Pierre AMBS	Université de Haute-Alsace
Rapporteurs	Nadine LE FORT–PIAT	ENSMM
	Youssoufi TOURÉ	Université d'Orléans
Examinateurs	Jean-Philippe URBAN	Université de Haute-Alsace
	Pierre PINO	Université de Metz
	Patrice WIRA	Université de Haute-Alsace

Thèse préparée au sein du laboratoire MIPS sous la direction de Jean-Philippe URBAN
École doctorale : « sciences : de la molécule aux matériaux et aux systèmes »

Approche neuromimétique modulaire pour la commande d'un système robot–vision

Gilles HERMANN

Thèse préparée au sein du laboratoire MIPS sous la direction de Jean-Philippe URBAN
École doctorale : « sciences : de la molécule aux matériaux et aux systèmes »

Remerciements

Ce mémoire met un terme à un travail de thèse qui a débuté il y a quatre ans. De nombreuses personnes ont participé au bon déroulement de cette période, aussi bien d'un point de vu personnel que professionnel. Ces quelques lignes leurs sont dédiées.

Je remercie principalement Maryline pour ses encouragements, son soutient et sa patience, surtout dans la période un peu plus délicate de la rédaction.

Je remercie également toute l'équipe du Trop et particulièrement: Jean-Philippe, mon directeur de thèse, Patrice avec qui se fut un plaisir de collaborer, Jean-Luc pour ses conseils avisés, M. Gresser pour l'accueil, Reda (mais où est-il?), Hubert notre McGyver, Djaffy pour son incommensurable culture Kabyle, Yohann qui répond rarement à mes interrogations et dont je n'ai pas de réponses à ses questions, Laurentiu, Gauthier ou encore François.

Je remercie les professeurs Nadine PIAT, Youssouffi TOURÉ et Pierre AMBS ainsi que Pierre PINO d'avoir bien voulu être membres de Jury. Leurs remarques, souvent intéressantes, seront prises en compte pour les développements futurs de ces travaux.

Enfin, sur un plan plus personnel, je remercie famille, belle-famille, amis. Notamment Bruno.

Table des matières

Table des figures

Introduction

Les robots que nous trouvons dans les industries de montage sont encore loin de se comporter comme celui que nous pouvons voir dans la publicité de la Xsara Picasso. Celui-ci peint la voiture aux chevrons au gré de ses envies et change radicalement de comportement à la vue d'un agent de maintenance.

Bien au contraire, les robots manipulateurs, comme ceux de l'industrie automobile par exemple, ont des rôles bien définis. Ils exécutent inlassablement les mêmes mouvements, souvent avec précision, dans un environnement parfaitement connu, où les conditions de fonctionnement restent les mêmes. Ils sont généralement programmés pour une tâche précise et réglée manuellement par des opérateurs expérimentés. Le moindre changement de paramètres dans la scène robotique n'est pas automatiquement pris en compte et le fonctionnement du robot n'est alors plus assuré. Il n'y a pas d'interactions entre le robot et son environnement.

Depuis quelques années, les progrès technologiques dans de nombreux domaines ont permis d'équiper les robots de capteurs extéroceptifs. Ces capteurs permettent de renseigner sur l'environnement et augmentent ainsi l'interaction avec celui-ci. L'un de ces capteurs est la caméra vidéo. Elle permet de fournir une importante quantité d'informations à travers lesquelles le robot va pouvoir percevoir son environnement, le comprendre et réagir avec, de manière autonome.

L'introduction de la vision dans un système robotique, ne va pas sans soulever un certain nombre de problèmes. Les images fournies par la caméra ou les caméras, dans un système de stéréo-vision, nécessitent une phase de traitement qui permet d'extraire les informations utiles au calcul d'une commande, pour une tâche donnée. Généralement coûteux en temps de calculs, les traitements d'images sont l'objet d'études dédiées. La réduction du temps de ces traitements est importante pour qu'ils

puissent être utilisés en ligne, c'est-à-dire à la cadence vidéo. Pour cela, quelques primitives sont extraites des images. Elles doivent être pertinentes et représentatives de la scène en évolution.

L'asservissement visuel, la science qui permet de contrôler les mouvements d'un robot en utilisant les informations images, se base sur le calcul d'une erreur entre une pose (position et orientation) effective et une pose à atteindre. Cette boucle de retour tente de formaliser la relation *sensori-motrice*, c'est-à-dire la relation qui existe entre l'espace des caractéristiques visuelles, et l'espace de commande du robot.

Cette relation entre les deux espaces est complexe, non-linéaire et dépendante de nombreux paramètres. Elle peut par exemple, correspondre au modèle inverse du système de vision couplé au modèle inverse d'un bras robotique. Ces modèles peuvent être établis de manière théorique, ils nécessitent alors l'emploi des modèles des différents éléments du système de vision et du système robotique ainsi que d'une phase d'étalonnage pour déterminer certains paramètres. La qualité de l'asservissement visuel va directement dépendre de la qualité des modèles utilisés. L'approche neuronale peut être considérée comme une alternative à l'approche modèle.

Les techniques neuronales permettent d'apprendre des modèles de connaissances. La réponse d'un réseau de neurones à une entrée est conditionnée par les poids synaptiques, qui correspondent à la mémoire du réseau. L'apprentissage d'un modèle est effectué à partir d'observations faites du système à mimer. Si l'apprentissage est permanent, il permet d'adapter les modèles estimés aux changements de configuration de la scène robotique.

Une zone d'ombre plane cependant sur les réseaux de neurones. Leurs performances sont tributaires de la taille des réseaux et de la dimentionnalité du vecteur d'entrées. Face à ces limitations, nous cherchons à concevoir des architectures modulaires, c'est-à-dire des architectures composées de structures neuronales. Les architectures modulaires peuvent être de différentes natures et sont notamment capables de décomposer une tâche complexe en plusieurs sous-tâches plus faciles à mettre en œuvre. L'objectif de la modularité est de développer des structures neuronales de taille raisonnable lorsque la complexité d'une tâche augmente, afin d'en permettre l'apprentissage.

Lorsque les structures neuronales sont appliquées à l'asservissement visuel, celui-ci devient adaptatif ou par apprentissage.

Contribution de la thèse

L'asservissement visuel d'un système robotique fait intervenir des modèles complexes dépendants de nombreuses grandeurs. L'apprentissage de tels modèles avec des ré-

seaux uniques n'est pas réellement facile. Le grand nombre de paramètres d'entrées fait rapidement chuter l'efficacité des algorithmes neuromimétiques.

Ce travail de thèse fait suite à une première thèse soutenue en 1999 (Buessler, 1999). Il s'étend à l'apprentissage de fonctions robotiques 3D complexes.

Dans cette thèse, nous nous sommes intéressés à la mise en place de structures neuronales capables d'apprendre des tâches robotiques complexes. Les structures que nous avons choisies utilisent une décomposition séquentielle de la tâche à accomplir. Chacun des modules d'une structure séquentielle ne traite qu'une partie des informations disponibles. Par conséquent, les modules de la structure modulaire sont de dimensionnalité réduite par rapport à la dimensionnalité globale du problème de départ.

Outre le fait de réduire la taille des réseaux utilisés, la modularité permet aussi de rendre l'apprentissage plus efficace et plus rapide. En effet, une décomposition découle d'une étude préalable du système à mimer, elle permet donc d'introduire des connaissances a priori dans la structure. La décomposition doit rester néanmoins la plus simple possible.

Le fait de décomposer une tâche en sous-tâches introduit inévitablement des grandeurs intermédiaires entre les différents modules. Ces grandeurs sont nécessaire pour pouvoir entraîner chacun des modules. Comme celles-ci ne sont pas mesurables, il faut mettre en œuvre de nouvelles structures d'apprentissage qui permettent de fournir des exemples d'apprentissages à chacun des modules.

L'introduction de nouveaux modules dans le flux inverse permet de faire des estimations des grandeurs intermédiaires. Elles vont servir de sorties désirées à chacun des modules. L'architecture modulaire devient bidirectionnelle. Une telle architecture est montrée sur la figure 1.

Cette figure montre la décomposition d'une fonction complexe M en deux modules M1 et M2. Chacun des deux modules traite une partie du vecteur d'entrées $[\mathbf{x_1} \quad \mathbf{x_2}]$. La dimensionnalité des modules M1 et M2 est donc plus petite que celle du module M. Cette décomposition introduit une variable intermédiaire $\hat{\mathbf{z}_1}$, qui est la réponse du module M1. Un module $M2^{inv}$ est ajouté à la structure, dans le flux inverse. Il fait une seconde estimation $\hat{\mathbf{z}_2}$ de la variable intermédiaire. Cette estimation sert de

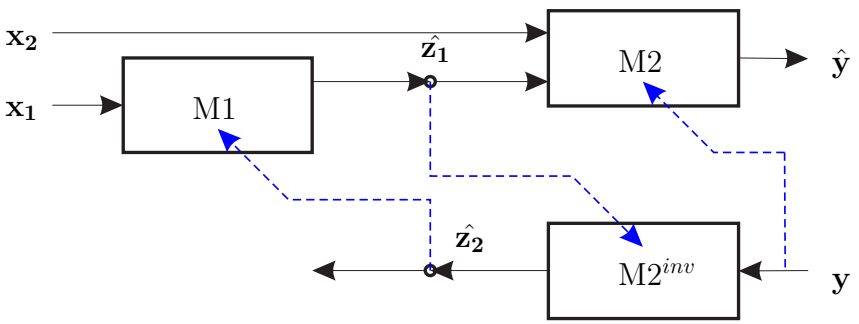


Fig. 1 *Exemple d'architecture modulaire bi-directionnelle*

modèle d'apprentissage au module M1. De la même manière, l'estimation $\hat{\mathbf{z}}_1$ sert de modèle pour l'apprentissage de $M2^{inv}$.

La modularité permet de simplifier le traitement d'une tâche complexe. Nous allons montrer dans cette thèse de quelle manière il est possible d'entraîner une telle structure et donc d'apprendre un système complexe.

Organisation du mémoire

Ce mémoire est divisé en quatre chapitres :

Chapitre 1 : Asservissement visuel

Ce premier chapitre traite de l'asservissement visuel en robotique. L'intégration de capteurs visuels dans la boucle de commande, permet au robot de percevoir son environnement et d'interagir avec celui-ci. La relation qui lie l'espace du capteur à l'espace du robot peut être soit modélisée, soit apprise. C'est cette seconde approche qui nous intéresse plus particulièrement, on parle alors d'asservissement visuel adaptatif.

Nous passerons en revue les différents principes de l'asservissement visuel ainsi que le principe de la commande avant de nous focaliser sur les techniques d'asservissement visuel adaptatifs.

Chapitre 2 : Connexionnisme et modularité

Deux thèmes sont abordés dans ce chapitre. Le premier concerne le connexionnisme. Nous y présenterons différentes architectures neuronales ainsi que la notion d'apprentissage. Nous parlerons ensuite du connexionnisme en robotique, en présentant notre plate-forme expérimentale ainsi que les différentes contributions du laboratoire dans ce domaine. Les réseaux de neurones que nous utiliserons dans nos architectures modulaires, sont des cartes auto-organisatrices. Elles seront présentées.

Le second thème abordé dans ce chapitre concerne la modularité. Nous présenterons différentes variantes de modularité, ainsi que les notions d'apprentissage liées à celle-ci. Nous parlerons ensuite de flux inverse et d'architecture bidirectionnelle, qui composent une structure d'aide à l'apprentissage, indispensable pour entraîner les modules d'une architecture séquentielle. Nous illustrerons ce chapitre par diverses contributions utilisant la modularité dans des applications robotiques.

Chapitre 3 : Application au robot trois axes

Dans une première application, nous allons utiliser les informations provenant de deux caméras orientables et déportées pour estimer les positions angulaires d'un bras robotique à trois degrés de liberté. Le nombre important de variables intervenant dans cette estimation nous conduit à utiliser une structure modulaire bidirectionnelle.

Ce chapitre présente l'environnement de travail utilisé ainsi que les résultats de l'estimation des positions angulaires obtenus respectivement avec une architecture classique puis avec une architecture modulaire bidirectionnelle. Une discussion relative à cet apprentissage clôturera ce chapitre.

Chapitre 4 : Application au robot quatre axes

La seconde application traitée dans ce dernier chapitre concerne l'estimation de l'orientation de l'effecteur d'un robot quatre axes. Les informations visuelles utilisées sont issues d'une tête robotique et sont les positions de l'effecteur et du poignet dans les images, ainsi que les orientations des deux caméras.

Une analyse de la géométrie du problème nous permet de dégager une architecture modulaire. Nous développerons les différents modèles qui découlent de la décomposition, ceux-ci vont servir de référence pour valider notre approche. Les résultats de l'apprentissage seront présentés en fin de chapitre et seront suivis d'une discussion.

1

Asservissement visuel

Sommaire

Bien longtemps, les robots industriels, ou les bras manipulateurs, étaient contraints d'effectuer des tâches simples, répétitives et bien sûr prédéterminées. Ceci était dû au fait qu'ils ne pouvaient pas percevoir leur environnement. Ce manque peut avoir des conséquences désastreuses sur la réalisation d'une tâche puisqu'il est impossible de prendre en compte les changements qui peuvent survenir comme une modification de l'environnement par exemple.

L'arrivée de nouveaux capteurs, tels que les caméras, permettent aux robots de mieux percevoir l'environnement. Ils peuvent se localiser par rapport aux autres éléments qui composent la scène. Ils peuvent tenir compte des changements de l'espace. Les asservissements visuels permettent de commander les déplacements du robot en utilisant les mesures effectuées dans les images.

Différents points vont être abordés au cours de ce chapitre. Dans un premier temps, nous allons présenter quelques récentes avancées dans le domaine de l'asservissement visuel, notamment en robotique médicale, en réalité virtuelle ou encore en réalité augmentée.

Nous parlerons ensuite brièvement de l'extraction d'informations dans les images. Étape importante, puisqu'elle permet d'extraire des mesures qui servent à déterminer la commande. De plus, ces traitements doivent être rapides afin de permettre leur utilisation en ligne. Nous en profiterons pour présenter quelques méthodes qui ont été mises en œuvre au laboratoire.

Différentes options sont possibles en asservissement visuel. La caméra peut être embarquée ou déportée, l'erreur qui sert à la commande peut être déterminée dans l'espace capteur, ou dans l'espace 3D, etc. Tous ces éléments seront abordés dans une partie traitant des principes de l'asservissement.

Nous allons ensuite présenter le principe de la commande en introduisant la notion de jacobienne, matrice qui permet de lier une différence dans l'espace d'entrées à une différence dans l'espace de sorties.

La matrice jacobienne peut être établie en utilisant les modèles de différents éléments, ou bien définie par apprentissage. Nous présenterons ce dernier cas avant de conclure, en citant un certain nombre de contributions dans le domaine.

1.1 Généralités et récentes avancées

L'insertion d'informations visuelles, issues d'une ou de plusieurs caméras, dans une boucle de commande permet de contrôler le mouvement d'un robot. Les scènes à considérer ne sont plus simplement statiques. En effet, il est possible de se positionner sur une cible mobile, suivre une trajectoire, la modifier en cours de route en cas d'obstacles, etc. L'ajout de la perception permet d'améliorer l'action et d'augmenter la flexibilité ainsi que la précision.

Avant d'aller plus loin dans ce chapitre et présenter l'extraction d'informations, les principes de l'asservissement visuel, les principes de la commande robotique et l'asservissement visuel adaptatif, nous allons présenter succinctement quelques applications récentes pour lesquelles les techniques d'asservissement visuel sont utilisées.

Depuis quelques années la robotique médicale connaît des développements importants et des succès spectaculaires (Zanne, 2003 ; Chaumette, 2003). On trouve notamment une application d'aide au geste chirurgical par asservissement visuel. Le système qui a été développé permet de conserver l'instrument dans le champ visuel du chirurgien, ou de positionner l'instrument automatiquement à une distance prédéfinie. Les principales difficultés d'une telle application sont apportées par l'environnement, peu structuré où la détection d'indices est difficile, et par le fait que les outils passent par un point d'incision, ce qui contraint les mouvements des robots. Des travaux sont en cours pour assister le chirurgien dans d'autres gestes médicaux (comme la suture par exemple). Une seconde application en médecine consiste à compenser les mouvements des organes induits par la respiration ou le rythme cardiaque. Ainsi ces mouvements deviennent transparents pour le chirurgien qui télé-manipule. Dans ce cas, un asservissement visuel GPC (commande prédictive généralisées) est utilisé. Enfin, pour en terminer avec la robotique médicale, citons encore les travaux entrepris pour la réalisation de procédures percutanées en radiologie interventionnelle par imagerie scanner. Cela consiste à enfoncer une aiguille dans des mini-tumeurs du foie qui ne sont visibles qu'à l'imagerie scanner, afin de les détruire ensuite par radio-fréquence. Une telle procédure permet d'être beaucoup plus précis et permet de ne plus exposer le chirurgien aux rayonnements X.

L'asservissement visuel est présent dans d'autres domaines d'applications que celui de la robotique. Nous pouvons citer, par exemple, la réalité virtuelle et la réalité augmentée. Dans le premier cas, on trouvera notamment le contrôle du regard d'un humanoïde virtuel. Il est également possible de générer par asservissement visuel des mouvements spécialisés de type cinématographie pour la caméra virtuelle de restitution, ou encore de contrôler les mouvements de cette caméra en évitant les obstacles et les occultations (technique utilisée entre autres pour la visite de musées virtuels). Dans le second cas, celui de la réalité augmentée, on trouvera par exemple, le contrôle des mouvements d'une caméra virtuelle afin de minimiser l'erreur entre l'image observée par une caméra réelle et la projection du modèle CAO des objets

sur la vue de la caméra virtuelle. La position de la caméra virtuelle minimisant cette erreur, obtenue à la convergence de l'asservissement visuel « virtuel », fournit la pose de la caméra réelle. L'application au suivi 3D et à la réalité augmentée est alors immédiate (Comport *et al.*, 2003).

Nous allons dans la section suivante, présenter brièvement l'extraction d'informations dans les images.

1.2 L'extraction d'informations

α.– Principe

Un capteur de vision fournit une importante quantité d'informations relatives à la scène. Les traitements d'images permettent alors d'en isoler, d'en extraire quelques une utiles à l'application. En effet, le traitement est propre à chaque tâche.

Lors de la mise en œuvre d'une application robotique, nous sommes rapidement confrontés au choix des informations à prendre en compte et donc à la manière de les traiter. Plus l'image est complexe, plus les temps de traitement seront grands pour en extraire l'information utile. C'est pourquoi, on peut trouver dans les applications robot-vision des systèmes simplifiés, ou bien des scènes conçues de telle manière à en facilité l'interprétation. Le traitement des informations doit être rapide pour les utiliser dans un système temps-réel.

Les premières réalisations qui ont été mises en œuvre sur la plate-forme expérimentale du laboratoire utilise un traitement d'image relativement simple. Le seul objet à identifier dans les images est l'effecteur du bras robotique. Celui-ci à été équipé d'une petite ampoule afin de faciliter l'observation. Pour déterminer la position de cet effecteur dans les images, trois étapes sont nécessaires : une binarisation, un seuillage et un calcul de centre de gravité. Pour accélérer le traitement, il est possible de traiter des régions d'intérêts (ROI) plutôt que de traiter l'image dans sa globalité. Des techniques prédictives ont été mise en place déterminer les positions de ces ROI (Wira, 2002 ; Hermann, 2000).

Des tâches plus complexes imposent d'utiliser des algorithmes plus performants et plus robustes. En effet, les nouvelles applications demandent l'identification de plusieurs éléments, comme l'effecteur, le poignet, les cibles, des obstacles, etc. L'introduction de la couleur peut être envisagée comme une solution (Swain et Ballard, 1991).

β.– L'utilisation de la couleur

Jusqu'à présent, les travaux de l'équipe se focalisaient sur l'apprentissage d'algorithmes neuronaux pour des tâches de positionnement d'un l'effecteur en utilisant les informations visuelles. Par conséquent, une importance moindre était portée à l'environnement, c'est pourquoi celui-ci était conditionné de telle sorte que le seul élément observé dans la scène était l'effecteur du bras robotique.

Nous nous intéressons aujourd'hui au contrôle du bras dans l'environnement réel (Buessler *et al.*, 2004 ; Hermann *et al.*, 2004a). Une première application consiste à lui faire atteindre une cible connue. La caméra est montée sur l'effecteur du bras et délivre des images d'objets connus, qui peuvent être vus sous différents angles, à différentes distances et dans diverses conditions d'illumination. Le système de vision doit identifier et localiser un objet donné en référence, et le suivre le temps que le bras effectue le mouvement.

Les techniques choisies permettent une utilisation en ligne (à la cadence vidéo) et sont basées sur la similarité d'histogrammes de couleurs.

1.3 Principes de l'asservissement visuel

α.– Généralités sur l'asservissement visuel

Qu'est-ce que l'asservissement visuel ?

Les techniques d'asservissement visuel consistent à utiliser les informations fournies par une ou plusieurs caméras afin de contrôler les mouvements d'un système robotique. Les capteurs visuels viennent s'ajouter aux capteurs proprioceptifs qui diagnostiquent l'état interne du robot (position et vitesse articulaires, couples d'efforts, etc.). Ces capteurs, extéroceptifs, renseignent sur l'état du robot, en tenant compte de l'environnement (position des cibles, d'obstacles, etc.).

Positions du capteur visuel

Les configurations du capteur visuel peuvent aller d'une caméra embarquée sur l'effecteur du robot à plusieurs caméras déportées ou montées sur un autre système robotisé. La première configuration, dite *eye in hand* est la plus couramment utilisée (voir figure 1.1a)). La relation connue et constante entre la position de la caméra et celle de l'effecteur simplifie grandement la tâche d'asservissement. La seconde configuration permet de visualiser les positions angulaires du bras robotique. Dans le cas où plusieurs objets sont observés, comme par exemple la cible et l'effecteur

(voir figure 1.1b)), les temps de calculs des traitements d'images deviennent plus contraignants.

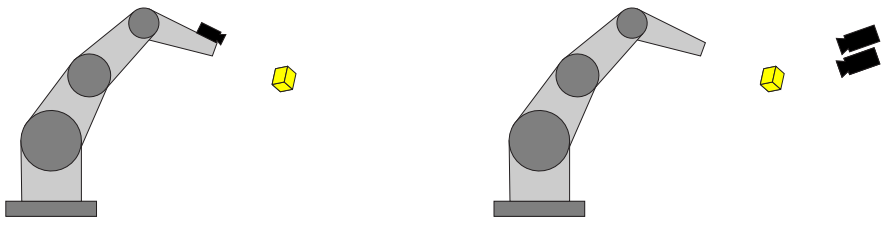
Asservissement statique et asservissement dynamique

L'approche asservissement visuel a été introduite dans les années 80 par Sanderson et Weiss (Sanderson et Weiss, 1980 ; Weiss, 1984). Ils proposent une approche formelle pour l'analyse des structures des commandes visuelles. Depuis, de nombreux travaux ont été menés. Nous pouvons citer, par exemple, les travaux de Feddema et Mitchell (1989) ; Urban (1990) ; Chaumette (1990), où des expérimentations ont été menées sur des plates-formes expérimentales, en parallèle avec des apports théoriques.

Nous pouvons distinguer deux styles d'asservissement visuel : statique ou dynamique. Dans le premier cas, l'erreur est évaluée lorsque la scène n'évolue plus. Cet asservissement, dit *static look and move*(Weiss, 1984), se déroule en trois étapes : (i) le système de vision regarde la scène, (ii) la commande est déterminée, (iii) le robot exécute le mouvement. La stabilité de cette approche lui confère de nombreuses applications industrielles (Hutchinson *et al.*, 1996).

Dans le second cas, les trois étapes citées précédemment ne sont plus exécutées les unes après les autres, mais sont exécutées simultanément. L'erreur de positionnement est évaluée en permanence (au fur et à mesure que les informations sont extraites des images), ce qui permet de corriger la commande du robot en temps réel. Par opposition au premier cas, cet asservissement dynamique est qualifié de *dynamic look and move*. Nous pouvons trouver des mises en œuvres de cette approche dans Hashimoto (1993), un recueil d'articles à propos de l'asservissement visuel, avec notamment un compte-rendu des avancées dans le domaine par Corke (1993). On trouvera des applications plus récentes dans IEEE RA (1996) ou encore dans Martinet (1999). Il est intéressant de noter l'essor de la robotique et de l'asservissement visuel dans les applications médicales (Krupa, 2003 ; IEEE RA, 2003).

Jusqu'ici, nous n'avons pas donné plus d'informations concernant la vision, mis à part qu'elle fournit les informations nécessaires à la commande. Des connaissances plus



a) b)

Fig. 1.1 *Configurations bras / caméra*

La figure a) correspond à la configuration eye in hand. La figure b) représente une situation où les caméras sont déportées. Elles peuvent observer simultanément la cible et l'effecteur

riches de l'environnement peuvent être obtenues si le capteur visuel est rendu actif (Chaumette, 1998), on parle alors de vision active. En vision active, les caméras sont indépendantes, et sont capables de modifier leur configuration afin de s'adapter au mieux aux changements de la scène. En vision active, les caméras peuvent ajuster leur mise au point, zoomer sur des zones d'intérêts, s'orienter, voire se déplacer (Aloimonos, 1993 ; Malis, 2004).

β.– Les techniques d'asservissement visuel

Les images fournissent beaucoup d'informations. Il faut alors sélectionner au mieux, un ensemble de mesures permettant de contrôler les degrés de liberté souhaités. Une loi de commande doit être élaborée afin que les mesures atteignent une valeur désirée ou suivent une trajectoire spécifiée. Le principe de la commande consiste en la régulation d'une erreur définie entre les mesures et les consignes.

Les informations fournies par une caméra sont à la base des informations 2D. En asservissement visuel, elles peuvent être utilisées ainsi (sous la forme de coordonnées de points, de paramètres de droites, etc.), ou alors être transformées par estimation ou par calcul, pour donner des mesures 3D.

L'une des premières problématiques en asservissement visuel, est de sélectionner, de construire et de combiner les informations visuelles qui permettent d'obtenir un comportement le plus proche possible de celui souhaité. Le choix des informations 2D ou 3D impose des lois de commande différentes. En effet, deux espaces de contrôle sont possibles : soit l'espace cartésien (celui du robot), soit celui du capteur (celui de l'image d'une caméra). Nous parlerons dans le premier cas d'asservissements 3D, et d'asservissements 2D dans le second cas. Quand il s'agit d'employer les deux types d'informations, nous parlerons d'asservissements hybrides.

Asservissement 3D

L'asservissement visuel 3D, ou *position-based visual servoing*, consiste à estimer dans l'espace 3D, à partir des informations images, la position de l'effecteur et la position de la cible. Le principe général est montré sur la figure 1.2.

Deux étapes successives composent cette approche. Une première étape de reconstruction consiste en l'estimation de la position de l'effecteur et de la cible dans un repère cartésien relié à l'espace tridimensionnel de la scène robotique. Différentes méthodes de reconstructions peuvent être utilisées. Elles sont basées sur :

- des techniques photogramétriques ;
- la triangulation ;
- sur l'obtention du mouvement et de la structure ;
- ou encore sur l'apprentissage.

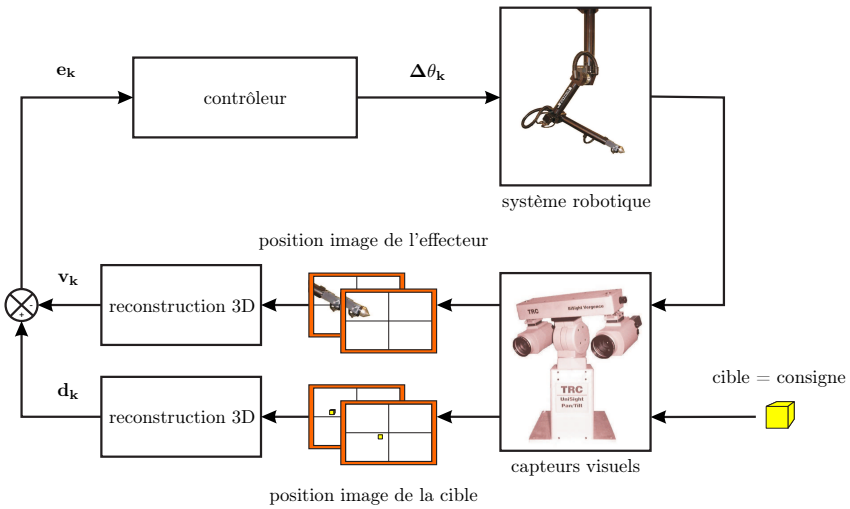


Fig. 1.2 *Principe de l'asservissement 3D*

L'asservissement 3D passe par une phase de reconstruction 3D des positions de l'effecteur et de la cible. L'erreur est évaluée dans l'espace de travail.

La seconde étape consiste à calculer la consigne. Elle est exprimée sous la forme d'un changement de repère lié à l'objet d'intérêt par rapport au repère lié à l'effecteur.

En théorie, il suffit d'une itération pour amener l'effecteur du robot sur sa cible. En pratique, cela n'est pas si simple. La qualité de la commande dépend bien évidemment de la qualité des reconstructions dans l'espace 3D. Les sources d'erreurs sont multiples. La première source d'erreurs est souvent liée à l'emploi d'un modèle de caméra trop simple, comme le modèle sténopé par exemple. Bien évidemment, des modèles plus sophistiqués existent. Ils intègrent par exemple des distorsions. Il reste cependant difficile d'obtenir un modèle exact de caméra. D'autres erreurs sont liées aux difficultés de calibrage du système visuel, ou encore aux bruits de mesures issues des images.

Les effets de ces erreurs de modèle et/ou de calibrage peuvent être en partie limitée par la mise en œuvre d'un rebouclage et d'un asservissement en vitesse. Ainsi, la stabilité du système est assurée.

Malgré ces quelques inconvénients, il faut noter qu'en asservissement visuel 3D, l'espace de contrôle reste équivalent à l'espace opérationnel du robot. Les lois de commande restent par conséquent assez simples.

Nous pourrons trouver un exemple récent de reconstruction 3D dans Alhaj *et al.* (2003) pour une tâche de positionnement. Les auteurs associent une reconstruction 3D à un asservissement visuel 2D.

Asservissement 2D

Une autre approche, dite asservissement 2D ou *image-based visual servoing*, définit la commande dans l'espace des capteurs. C'est-à-dire dans les images dans le cas où

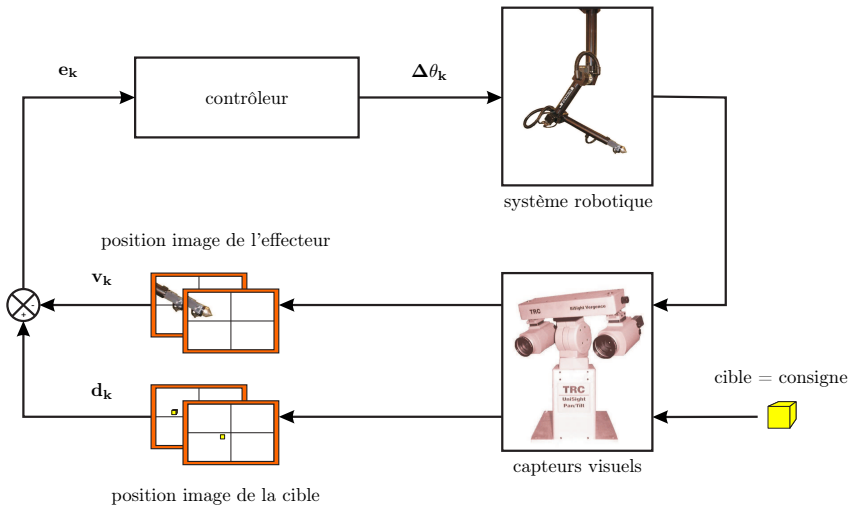


Fig. 1.3 *Principe de l'asservissement 2D*

L'asservissement 2D, ou direct, convertit directement une erreur évaluée dans l'espace sensoriel en une consigne dans l'espace de commande, sans passer par une étape de reconstruction.

les capteurs sont des caméras. L'erreur de positionnement mesurée dans les images, entre l'effecteur du robot et la cible à atteindre, est directement transformée en correction angulaire. Cette approche est illustrée par la figure 1.3.

Les informations images de la scène courante sont représentées par un vecteur s. L'asservissement visuel 2D consiste à amener cette représentation s à atteindre le motif s^*, qui représente la position finale souhaitée.

Le choix des informations visuelles à suivre est très important. Elles sont principalement de deux types : dynamiques ou géométriques.

Ce principe d'asservissement à le fort avantage d'être moins coûteux en temps de calcul, par rapport à un asservissement 3D, puisqu'il s'affranchit de l'étape de reconstruction 3D.

Citons par exemple Hamel et Mahony (2002), où les auteurs utilisent un asservissement visuel 2D pour la commande d'un système robotique. Dans cette application, la caméra est située sur l'élément à asservir.

Par contre, dans certains cas, comme une cible très éloignée, il peut arriver que la loi de commande diverge. Il faudra alors leur préférer des techniques d'asservissements hybrides.

Asservissement hybride

À la fin des années 90, une nouvelle approche d'asservissement visuel a été proposée par Malis et Chaumette (Malis, 1998). On la trouvera sous les noms d'asservissement 2D$\frac{1}{2}$ ou d'asservissement 2-$\frac{1}{2}$-D, lorsque les informations visuelles sont de type

géométriques (coordonnées de points, droites, etc.). Ou sous le nom d'asservissement d2-D/dt quand les informations visuelles sont du type vitesse (Chaumette, 1998).

Cette classe de schémas d'asservissement visuel *hybride* utilise à la fois des données 3D et des données 2D, mais sans nécessiter de connaître le modèle de la cible. Généralement, dans ces approches, la caméra est fixée sur l'effecteur du robot.

L'approche 2D$\frac{1}{2}$ est basée sur l'estimation de l'homographie qui relie l'image de plusieurs points à un motif désiré. À partir de l'homographie estimée, il est possible de calculer le déplacement en rotation que la caméra doit effectuer pour atteindre une position désirée, ainsi que son déplacement en translation à un facteur d'échelle près. Ce facteur d'échelle rend impossible l'emploi d'un asservissement 3D. La combinaison des informations 2D et 3D permet d'aboutir à une solution aux propriétés intéressantes.

Les avantages sont nombreux et significatifs (Malis et Chaumette, 2002). Il n'est plus nécessaire de connaître le modèle 3D des objets considérés car la localisation s'effectue directement à partir de la mise en correspondance entre l'image courante et l'image désirée.

Le comportement du système est satisfaisant du fait du découplage apporté entre les degrés de liberté en translation et ceux en rotation. Ce découplage a permis d'analyser la stabilité du système en présence d'erreur de calibration et d'établir les conditions analytiques sur les erreurs des paramètres intrinsèques et extrinsèques du capteur de vision pour assurer cette stabilité.

Cette contribution majeure trouve de nombreuses applications, dont en voici une liste non-exhaustive :

- Dans Mezouar et Chaumette (2002), la méthode présentée est basée sur l'approche des fonctions de potentiels, elle permet de considérer aussi bien des contraintes sur la trajectoire 3D du robot que des contraintes telles l'évitement des butées articulaires, l'évitement d'occultations de parties d'objets, ainsi que l'assurance que l'objet reste dans le champ de vision du capteur. L'absence de minima locaux n'est pas démontrée. La méthode semble donner des résultats satisfaisants en pratique.
- Dans Mezouar et Chaumette (2003), la méthode présentée est basée sur une commande optimale. Elle permet de calculer la forme analytique des trajectoires de points dans l'espace correspondant à un déplacement optimal de la caméra (translation en ligne droite, rotation selon une géodésique). Il semble impossible de déterminer des formes analytiques dans tous les cas (en rajoutant des contraintes par exemple, telle la conservation de la visibilité de l'objet). Cette méthode est actuellement améliorée en considérant une base d'images indexées.

- Dans Zanne (2003), la méthode présentée assure la visibilité de l'objet, même en présence d'erreurs de calibration de la caméra. Elle ne permet toutefois pas de prendre en considération d'autres contraintes.
- Dans Gangloff et Mathelin (2002) la modélisation complète d'une boucle d'asservissement visuel a été réalisée. La boucle comprend un robot manipulateur, une caméra et son système d'acquisition et de traitement d'images. Le modèle multivariable obtenu correspond à une linéarisation autour d'une configuration donnée du robot. Il tient compte des dynamiques du robot et du système d'acquisition d'images. Des techniques de commande prédictive généralisée (GPC) ont été employées pour contrôler la boucle. Elles possèdent l'avantage de garantir l'optimalité de la réponse pour chaque configuration du robot grâce à une technique de séquencement de gain. Deux tâches différentes ont permis de valider cette technique : le suivi rapide de cible (avec une caméra rapide, 240 images/sec.) et le suivi de profilé (Gangloff, 1999).

Ce principe d'asservissement visuel apporte une amélioration importante au niveau de la convergence et de la stabilité par rapport aux approches 2D et 3D. Citons encore Gans *et al.* (2002) qui comparent différents types d'asservissements 2D et d'asservissements hybrides

1.4 Principe de la commande

De manière tout à fait générale et simplifiée, les applications robotiques considérées consistent à déplacer l'effecteur d'un robot pour qu'il atteigne une cible, avec plus ou moins de contraintes sur la manière dont il doit effectuer cette tâche. L'asservissement visuel est alors basé, d'une part, sur la transformation inverse du système robotique et d'autre part sur la transformation inverse du système visuel. Bien que l'emploi des modèles inverses soit monnaie courante dans le domaine, leur établissement n'est néanmoins pas trivial. De plus, ceux-ci sont souvent liés à une scène et doivent, par conséquent, être modifiés à chaque changement de configuration. Pour élaborer une approche qui peut s'affranchir des modèles, nous nous tournons naturellement vers des systèmes à apprentissage.

Avant d'aller plus loin et de présenter l'asservissement visuel adaptatif, nous allons présenter quelques aspects de la commande, afin de bien comprendre les enjeux des contrôleurs et ce que peut apporter l'apprentissage.

α.– Modèles géométrique et cinématique

Pour la commande robotique, les contrôleurs peuvent utiliser des modèles de deux types : les modèles géométriques, ou les modèles cinématiques (encore appelés modèles différentiels)(Lallemand et Zeghloul, 1994 ; Coiffet, 1992).

Modèle géométrique

Le modèle géométrique est le modèle qui permet de lier deux espaces ensemble. Dans le cas du robot, les deux espaces à lier sont l'espace articulaire $\boldsymbol{\theta}$ et l'espace des tâches $\mathbf{x}$ (les positions dans l'espace tridimensionnel par exemple). En supposant le robot parfait, le modèle géométrique direct est :

$$\mathbf{x} = f(\boldsymbol{\theta}) \ , \tag{1.1}$$

avec $\boldsymbol{\theta} = [\theta_1,\theta_2,\ldots,\theta_m]^T$ et $\mathbf{x} = [x_1,x_2,\ldots,x_n]^T$. Ici, m correspond aux degrés de liberté du robot et n au nombre de composantes de $\mathbf{x}$ qui, lui, dépend de la tâche fixée (trois composantes pour une commande en position, six composantes pour une commande en position et orientation).

Si $f()$ représente la relation entre $\mathbf{x}$ et $\boldsymbol{\theta}$, alors le modèle géométrique inverse s'exprime de la manière suivante :

$$\boldsymbol{\theta} = f^{-1}(\mathbf{x}) \ . \tag{1.2}$$

Dans le cadre du contrôle d'un robot, c'est le modèle inverse qui est utilisé. Il faut définir le vecteur $\boldsymbol{\theta}$ qui permet au robot d'atteindre une position $\mathbf{x}$ (avec ou sans l'orientation). Comme la relation qui lie les deux espaces n'est pas bijective, il peut arriver qu'il n'y ait pas un vecteur solution $\boldsymbol{\theta}$ unique. Pour éviter ces redondances, nous sommes souvent amené à ajouter des contraintes géométriques supplémentaires.

Le modèle géométrique inverse reste une méthode assez peu performante et peu précise du fait de sa non-linéarité et de ses redondances. Les performances de la commande peuvent être accrues en utilisant le modèle cinématique.

Modèle cinématique

Les problèmes liés au modèle géométrique, tels que la non-linéarité de l'équation (1.1), ou alors le non-contrôle de la vitesse de déplacement du robot lors de l'exécution d'une tâche, peuvent être en partie résolus avec l'emploi du modèle cinématique.

Par abus de langage, nous emploierons indifféremment les termes de modèle cinématique et de modèle différentiel, alors qu'en fait le modèle cinématique découle d'une modélisation par les formules de la cinématique du solide, tandis que le modèle dif-

férentiel se déduit de la dérivation du modèle géométrique direct par l'opérateur de dérivation matriciel (il s'agit en fait d'une approximation, puisque nous ne tenons compte que du terme d'ordre un du développement de Taylor).

$$\Delta \mathbf{x} = \mathbf{J} \Delta \boldsymbol{\theta} \ . \tag{1.3}$$

Dans cette équation, $\mathbf{J}$ représente la matrice jacobienne du modèle géométrique dérivé. Cette relation lie les variations angulaires à des variations dans l'espace des tâches. De la même manière que précédemment, le modèle différentiel inverse peut être défini de la manière suivante :

$$\Delta \boldsymbol{\theta} = \mathbf{J}^{-1} \Delta \mathbf{x} \ . \tag{1.4}$$

Déterminer un modèle différentiel inverse $\mathbf{J}^{-1}$ revient en fait à un problème d'inversion de matrice. Sa complexité dépend du nombre de degrés de liberté.

Dans le cas d'un asservissement visuel, la matrice jacobienne $\mathbf{J}$ représente le modèle géométrique dérivé du système robotique et du système de vision. Elle lie un déplacement dans les images, à un déplacement angulaire.

β.– Approche différentielle et asservissement visuel

De manière générale, les méthodes d'asservissement visuel reposent sur une boucle de retour dynamique. Le point sensible dans cet asservissement est le calcul de la jacobienne inverse.

Si nous considérons qu'une cible et que l'effecteur du robot sont tous les deux visibles dans les images (en même temps), alors la commande est déterminée à partir de la différence du positionnement entre l'effecteur et la cible. Cette différence constitue l'erreur dans l'espace des images (cas d'un asservissement 2D, en position). Chaque élément présent dans les images peut être localisé par des coordonnées horizontales et verticales. Typiquement, ces informations sont fournies par un traitement d'images. Dans l'espace des images, l'effecteur est défini par le vecteur $\mathbf{v}_k$ et la cible par le vecteur $\mathbf{d}_k$. L'erreur peut alors s'exprimer de la manière suivante :

$$\mathbf{e}_k = \mathbf{d}_k - \mathbf{v}_k \tag{1.5}$$

Considérons la fonction vectorielle $f()$ qui réalise la transformation entre l'espace articulaire du robot et l'espace des caractéristiques des images. Nous avons :

$$\mathbf{v}_k = f(\boldsymbol{\theta}_k) \ , \tag{1.6}$$

avec $\boldsymbol{\theta}_k$ le vecteur contenant les valeurs de chaque articulation du robot, à l'instant k. La fonction $f()$ dépend aussi bien des paramètres du robot que des paramètres

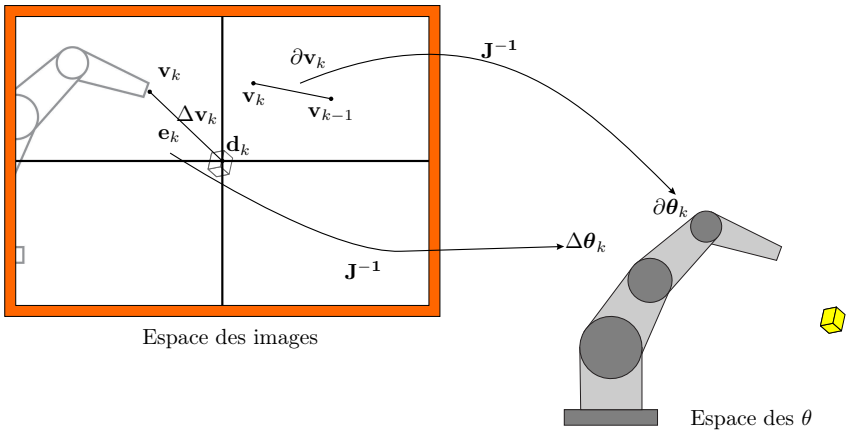


Fig. 1.4 *Principe de la commande*

La jacobienne $\mathbf{J}^{-1}$ *lie un déplacement* $\partial\mathbf{v}_k$ *à une variation* $\partial\boldsymbol{\theta}_k$*. En faisant une approximation linéaire au voisinage du point d'opération, Nous pouvons lier* $\Delta\mathbf{v}_k$ *à* $\Delta\boldsymbol{\theta}_k$*.*

du système de vision (en incluant la position relative des caméras par rapport au robot). La détermination de cette fonction, de manière mathématique, impose de connaître l'ensemble des paramètres liés au système robot–vision.

La matrice jacobienne $\mathbf{J}$ [1] de la fonction $f()$, lie, quant à elle, une petite modification $\partial\boldsymbol{\theta}_k$ dans l'espace angulaire, à une petite variation $\partial\mathbf{v}_k$ dans l'espace des images. Nous avons donc la relation suivante :

$$\partial\mathbf{v}_k = \mathbf{J}\partial\boldsymbol{\theta}_\mathbf{k} \ . \tag{1.7}$$

L'asservissement visuel s'appuie sur une approximation linéaire dans le voisinage du point d'opération. C'est-à-dire que nous considérons la même matrice $\mathbf{J}^{-1}$ pour lier deux déplacements proches dans les images, à deux variations angulaires. Les variations peuvent être mesurées entre deux instants. Nous avons alors : $\Delta\mathbf{v}_k = \mathbf{v}_k - \mathbf{v}_{k-1}$ et $\Delta\boldsymbol{\theta}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$. D'où :

$$\Delta\mathbf{v}_k = \mathbf{J}\Delta\boldsymbol{\theta}_k \ . \tag{1.8}$$

En prenant en compte ces considérations, il est alors possible de déterminer la variation angulaire à appliquer au robot, pour que celui-ci atteigne sa cible :

$$\Delta\boldsymbol{\theta}_k = \mathbf{J}^{-1}\mathbf{e}_k \ . \tag{1.9}$$

Ce principe est illustré sur la figure 1.4.

1. La matrice jacobienne $\mathbf{J}$ peut prendre les noms de matrice des sensibilités des caractéristiques images (Weiss, 1984), de matrice d'interaction (Chaumette, 1990) ou encore de matrice jacobienne image (Hutchinson *et al.*, 1996).

Plutôt que d'être calculée, la matrice inverse $\mathbf{J}^{-1}$ va être apprise. Des déplacements successifs vont servir d'exemples d'apprentissage. Dans ce cas, nous parlons d'asservissement visuel adaptatif.

γ.– Qualités de l'asservissement visuel

De manière générale, il est souhaitable qu'une loi de commande ait certaines qualités. En effet, on attend d'elle qu'elle soit stable de manière locale, voire globale, qu'elle soit robuste aux erreurs de mesure, aux erreurs de modélisation, qu'elle soit robuste par rapport aux informations visuelles dans l'image en tenant compte d'éventuelles occlusions par exemple. On peut souhaiter avoir des trajectoires de robot qui répondent à certains critères.

Parmi les développements récents qui permettent d'assurer la convergence, nous pouvons citer les travaux de Mezouar et Chaumette (2002, 2003). Les auteurs proposent une solution aux problèmes des grands déplacements. En effet, les approches classiques sont basées sur la régulation à zéro de l'erreur entre les valeurs courantes et désirées d'information visuelles, sélectionnées soit dans l'image, soit dans l'espace 3-D. Il est alors difficile d'introduire des contraintes sur la trajectoire réalisée et d'assurer la convergence quelle que soit la position initiale du robot. La solution que les auteurs proposent porte sur un couplage entre la planification de trajectoires et l'asservissement visuel.

Toujours dans l'optique de rendre robuste les tâches d'asservissement visuel, nous pouvons citer les travaux de Kragic (2001) ; Kragic et Christensen (2003), dans lesquels une étape de prédiction est mise en place.

1.5 Asservissement visuel adaptatif

α.– Principe

En asservissement visuel adaptatif (encore appelé asservissement visuel par apprentissage), la matrice inverse $\mathbf{J}^{-1}$ est estimée à l'aide de techniques itératives.

$$\hat{\mathbf{J}}_{k+1} = \hat{\mathbf{J}}_k + \Delta\mathbf{J}_k \ . \tag{1.10}$$

La correction $\Delta\mathbf{J}_k$ est généralement calculée par une règle de type LMS (*Least Mean Square*). Nous la retrouvons par exemple dans Jägersand (1996), où la matrice est adaptée sans être mémorisée, dans Ritter *et al.* (1992) ; Urban *et al.* (1998), où la notion de mémoire est associée à celle de l'adaptation, afin de constituer un véritable système d'apprentissage.

Dans Urban *et al.* (1998) l'espace d'entrées est partitionné en régions à l'aide d'une carte auto-organisatrice. Une matrice jacobienne est associée à chacune des régions et est apprise par un ADALINE. Cela permet de tenir compte du fait que l'estimation de la jacobienne est locale (vraie pour une petite région autour du point de fonctionnement). L'utilisation d'un tel réseau pour contrôler un robot qui a plus que trois axes devient difficile, surtout dans le cas d'une application temps réel. En revanche, les utiliser dans une structure adaptée (comme nous le verrons plus en avant dans ce mémoire) permet de traiter des cas bien plus complexes.

L'asservissement visuel adaptatif peut aussi s'appliquer au capteur visuel lui-même. Dans Berthouze *et al.* (1996) le contrôle de la boucle de vision active se fait par apprentissage.

β.– Quelques exemples en asservissement visuel adaptatif

Nous allons, pour terminer ce chapitre, présenter quelques applications faisant intervenir l'asservissement visuel adaptatif. Elles sont présentées dans quelques catégories telles que l'asservissements 2D, l'asservissements 3D, les systèmes non-calibrés ou encore les architectures modulaires. Notons que le fait qu'une contribution soit mise dans une catégorie ne l'exclue pas de toutes les autres, en effet, une contribution en asservissement 2D peut très bien utiliser une architecture modulaire.

Asservissements 2D

Hosoda et Asada (1994) proposent une méthode d'estimation de la jacobienne qui relie les variations des signaux 2D aux mouvements d'un robot. Cette estimation ne nécessite pas de connaissances a priori du système caméra-robot. Cette étude porte sur trois degrés de liberté.

Quant à lui, Jägersand (1996) utilise un algorithme de type LMS pour la correction en ligne de la matrice jacobienne et montre son efficacité pour le contrôle de robot dans des situations requérant une grande précision. Dans Kuhn (1997), nous retrouvons l'algorithme LMS auquel est ajouté une notion de mémorisation.

De nombreux asservissements visuels adaptatifs utilisent des réseaux de neurones pour faire une estimation de la jacobienne. Citons par exemple Miller III (1989) ; Miller III *et al.* (1990), qui utilisent un réseau de type CMAC (*Cerebellar Model Articulation Controller*) et élaborent une méthode d'apprentissage rapide pour le contrôle en boucle fermée qu'ils adaptent à différente tâche robotique. Ritter et Schulten (1986, 1988) ; Ritter *et al.* (1992) qui utilisent des cartes auto-organisatrices pour apprendre le modèle cinématique d'un robot trois axes.

Asservissements 3D

Les travaux de Breton (1999) présentent un asservissement 3D dédié à la coordination sensori-motrice d'un système de vision binoculaire et d'un système robotique indépendant. Aucune connaissance a priori de la géométrie du système robot-vision n'est requise. De même, aucun système d'étalonnage n'est nécessaire. La structure neuromimétique qui est utilisée pour l'apprentissage comporte deux modules séquentiels. Un premier fait une reconstruction de l'espace 3D en utilisant des techniques statistiques du traitement du signal. Le second s'occupe de la commande robotique. Il utilise un réseau HYPSOM (*HYPerplan-based Self-Organizing Map*)

Systèmes non-calibrés

Un avantage certain dans l'emploi de techniques adaptatives est qu'il est possible de se passer de la phase de calibration du système visuel. Dans Piepmeier *et al.* (2004) par exemple, les auteurs utilisent l'algorithme des moindres carrés récursifs pour estimer la matrice jacobienne. Ils montrent par ailleurs, que leur méthode quasi-Newton dynamique permet de s'affranchir de la phase de calibration.

Il reste cependant vrai que certaines approches classiques se passent aussi de cette phase de la calibration, comme par exemple dans Li et Lu (2004), ou alors font appel à des techniques d'auto-calibration, comme dans Ji et Dai (2004) par exemple.

Architectures modulaires

Les architectures modulaires sont généralement des structures qui combinent plusieurs modules neuronaux. Nous reviendrons plus en détail sur ces structures dans le chapitre suivant. Citons tout de même les travaux de Miyamoto *et al.* (1996) ; Miyamoto et Kawato (1998), où les auteurs font apprendre le jeu de Kendema ou encore apprendre à servir à un robot, ou encore les travaux de Wolpert et Kawato (1998), où l'architecture est composée de plusieurs paires de modèles inverses et de modèle directs, pour le contrôle robotique.

De nombreux travaux menés au laboratoire utilisent des architectures modulaires. Citons par exemple Buessler *et al.* (1999) ; Buessler et Urban (1998) ; Buessler *et al.* (2002). Nos travaux, présentés dans les chapitres suivants, utilisent ce principe (Hermann *et al.*, 2003b, 2004c). Dans ces travaux, les architectures modulaires sont utilisées afin de réduire les dimensions des réseaux de neurones. Cela facilite l'apprentissage de tâches complexes et le rend plus robuste.

1.6 Conclusion

Ce premier chapitre nous a permis de nous familiariser avec l'asservissement visuel et de montrer à quel point il est important de doter les systèmes robotiques de la vision, afin qu'ils puissent interagir avec leur environnement.

Les capteurs visuels utilisés fournissent des images riches en informations. Il est alors important d'utiliser le traitement adéquat, qui va permettre d'extraire les informations utiles à la commande, pour une tâche souhaitée. Nous avons brièvement présenté la contribution du laboratoire dans ce domaine.

L'asservissement visuel peut être abordé de diverses manières. En effet, les traitements sont différents à partir du moment où une ou plusieurs caméras sont placées soit sur l'effecteur, soit déportées, si l'erreur qui sert à calculer la commande est définie dans les images (espace des capteurs) ou alors dans l'espace du robot, etc. Nous avons regroupé ces considérations dans les principes de l'asservissement visuel.

En asservissement visuel, ce sont les informations issues de capteurs tels que des caméras, qui permettent de définir l'erreur utilisée pour calculer une commande. La matrice d'interaction, qui permet de passer de l'erreur à la commande, résulte de divers modèles tels que celui du système robotique, ou celui du système visuel.

Cette matrice jacobienne peut être déterminée de manière mathématique ou en utilisant des méthodes adaptatives, notamment avec des réseaux de neurones. C'est cette dernière approche que nous avons retenue.

Dans le chapitre suivant, nous allons aborder deux thématiques. Dans un premier temps, nous présenterons les réseaux de neurones et le connexionnisme en robotique. Dans un deuxième temps, nous présenterons les structures neuronales modulaires de manière générale, puis liées à la robotique.

2

Connexionnisme et modularité

Sommaire

PAR *défi, par besoin, par soif de savoir, ou même par jalousie peut être, l'Homme cherche sans cesse à comprendre ou à recopier ce que Dame-Nature fait si bien.*

Le connexionnisme, par exemple, est né de cette envie de reproduire le cerveau humain, ou du moins de reproduire ses capacités d'apprentissage. En effet, n'est-il pas merveilleux de constater à quel point il nous est facile de nous adapter à notre environnement ? Tout cela a été acquis par apprentissage, par mimétisme. L'empreinte de la biologie est très présente dans le connexionnisme. Il suffit de regarder les termes qui y sont employés. Nous parlons de réseaux de neurones, de poids synaptiques, de colonne corticale, etc. Si l'on voulait pousser le parallèle jusqu'à ce mémoire, on pourrait dire que le premier chapitre concerne les sens, la perception, une manière d'acquérir des informations, alors que le second chapitre concerne plutôt la mémoire et la capacité d'apprentissage.

En fait, ce chapitre va nous permettre de mettre en place un certain nombre d'outils pour l'élaboration de contrôleurs neuronaux performants dédiés à des tâches robotiques complexes.

Deux thèmes vont être développés. Nous présenterons dans un premier temps les réseaux de neurones (architectures, paradigme d'apprentissage, etc.) et dans un second temps, nous parlerons de modularité. Les architectures modulaires permettent de faire des compositions de réseaux, dans le but d'améliorer ou de simplifier l'apprentissage.

Chacun des deux thèmes sera illustré par des exemples d'applications que nous pouvons trouver dans le domaine de la robotique.

Enfin, nous expliquerons quels outils nous avons choisis, parmi cette palette, pour composer nos contrôleurs. La complexité des tâches que nous voulons effectuer requiert l'emploi d'une architecture séquentielle. De manière locale, cette architecture permet de réduire la dimentionnalité du problème. Chaque brique élémentaire sera composée de cartes auto-organisatrices.

Les règles d'apprentissage de ces architectures nécessitent d'utiliser des structures d'aide à l'apprentissage, telle que la bidirectionnalité.

2.1 Du neurone aux réseaux de neurones

Les réseaux de neurones artificiels sont apparus avec la description d'un modèle de neurone par McCulloch et Pitts en 1943. La première règle d'apprentissage est proposée par Hebb en 1950. Le Perceptron, proposé par Rosenblatt en 1962, est le premier modèle de réseau de neurones intégrant le neurone de McCulloch et la règle d'apprentissage de Hebb. Le véritable essor des réseaux de neurones a débuté au cours des années 80.

À l'instar de quelques outils mathématiques permettant de réaliser l'identification de fonctions à partir d'échantillons, les réseaux de neurones ont la capacité d'apprendre des fonctions non-linéaires arbitraires en utilisant un certain nombre d'exemples de la fonction. Les techniques neuromimétiques ont une formulation simple et obtiennent d'excellents résultats.

Les connaissances de chacun des neurones sont stockées dans les *poids* du réseau. Ces poids sont adaptés en fonction des entrées présentées, modifiant ainsi la réponse du réseau suite à une stimulation.

Les algorithmes neuronaux dépendent :

- de l'architecture, qui définit la façon dont la réponse est calculée et dont interviennent les poids ;
- du paradigme d'apprentissage : apprentissage non-supervisé, supervisé ou encore par renforcement.

α.– Les architectures neuronales

Les neurones qui composent les réseaux de neurones peuvent être liés entre eux de diverses manières. La capacité d'un réseau à apprendre un problème dépend de la taille et de son architecture. Bien que chaque type de réseau de neurones ait une architecture propre, il est possible de distinguer trois grandes familles : les réseaux unidirectionnels, les réseaux récurrents et une troisième catégorie qui regroupe tout ce qui ne rentre pas dans les deux premières. Une synthèse complète peut être trouvée dans le livre d'Haykin (1999) ou encore dans le livre de Dreyfus *et al.* (2002).

Les réseaux unidirectionnels

Les réseaux unidirectionnels (*feedforward networks* qui pourrait être traduit comme « alimenter et faire suivre ») ont une architecture de neurones organisés sous la forme d'une ou de plusieurs couches successives. Les sorties des neurones d'une couche deviennent les entrées des neurones des couches suivantes. Ces réseaux de neurones sont constitués d'une couche d'entrées qui n'effectue aucun traitement, d'une couche de sorties et d'une ou plusieurs couches cachées dans le cas de réseaux multicouches. Le Perceptron est un réseau qui adopte l'architecture d'un réseau

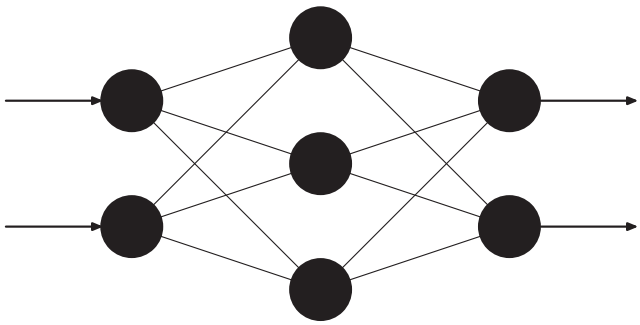


Fig. 2.1 *Réseau unidirectionnel*

Les cercles noirs représentent les neurones, un cercle noir avec une flèche entrante correspond à un neurone de la couche d'entrées, un cercle noir avec une flèche sortante correspond à un neurone de la couche de sorties, les lignes simples sont les interconnexions entre les neurones.

multicouche unidirectionnel. La figure 2.1 montre un schéma tout à fait général d'un réseau ayant cette architecture.

Les réseaux récurrents

Les réseaux récurrents désignent les réseaux dont chaque neurone est capable de communiquer avec tous les autres neurones de l'architecture. Ces réseaux sont monocouches ou multicouches et sont dotés de boucles de réactions internes. La figure 2.2 montre un réseau de ce type. Il est constitué de deux entrées et de deux sorties. Les délais introduits par les boucles de réaction font que ces réseaux présentent un comportement dynamique qui permet d'introduire explicitement la dimension temporelle. L'aboutissement est un processus de relaxation au cours duquel le réseau passe par une série d'états d'activation, l'état à l'instant $k+1$ étant calculé en fonction de l'état ou des états précédents. Le réseau se comporte comme un système dynamique tendant vers des états attracteurs, des cycles attracteurs ou des comportements chaotiques.

Les autres architectures

Dans ce dernier groupe, nous retrouvons tous les réseaux qui ne trouvent pas leur place dans les deux catégories précédemment citées.

En particulier, nous pouvons mentionner les réseaux à base de prototypes. Ces réseaux mémorisent un nombre limité d'exemples ou de prototypes représentatifs de chaque classe dans le cas de la classification, ou de parties de fonctions dans les cas d'approximations. Entrent dans cette catégorie les réseaux RBF, SOM, LVQ, ART, RCE, GAL, les réseaux probabilistes,... Cette liste n'est pas exhaustive. Les cartes auto-organisatrices seront plus largement abordées dans la suite de ce chapitre, de même que le concept de réseau de neurones modulaires.

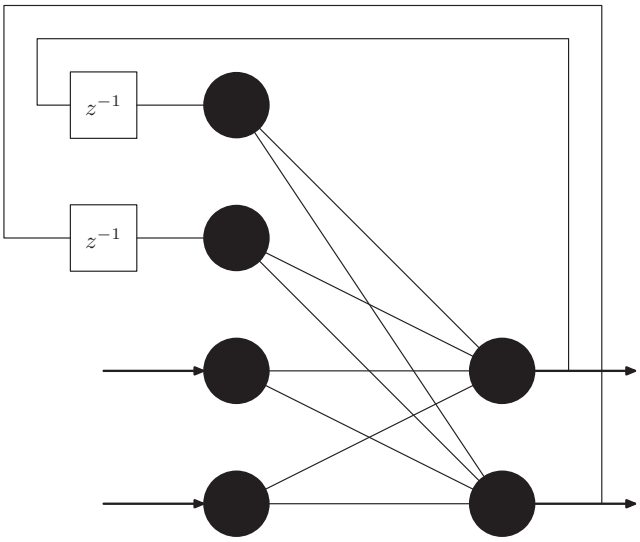


Fig. 2.2 *Réseau récurrent*

Les boites notées z^{-1} représentent des retards.

Pour conclure ce tour des architectures de réseaux de neurones, il faut encore parler des réseaux à architecture évolutive. Ils ont la particularité de subir un changement de structure au cours de la phase d'apprentissage. À La fin du cycle d'apprentissage, ces réseaux doivent aboutir à une structure optimale par rapport à la complexité du problème appris. Deux méthodes sont mises en œuvre pour faire évoluer l'architecture du réseau de neurones :

- Une première méthode d'accroissement (*growing network*) consiste à ajouter des neurones à une architecture, minimale à la base, à chaque fois que l'erreur d'apprentissage ne décroît plus. La difficulté réside dans le choix d'un bon critère d'ajout de neurones.
- Le pendant à cette première méthode est la méthode de simplification, de suppression, ou encore par élagage (*pruning network*). Cette fois-ci le réseau de neurones est sur-paramétré au départ. Des neurones sont ensuite supprimés au cours de l'apprentissage. Le critère de sélection des neurones à supprimer est plus facile à élaborer dans l'hypothèse où la suppression intervient lorsque le réseau a convergé.

β.– L'apprentissage

Pour qu'une architecture soit potentiellement capable de résoudre un problème, il faut déterminer une configuration de poids adaptée au problème. C'est le rôle du processus d'apprentissage. Il a pour but de trouver la solution optimale dans un espace à n-dimension. Il est possible d'obtenir une solution satisfaisante seulement si la complexité du réseau de neurones (le nombre de paramètres par exemple) est adaptée au problème à résoudre. Une complexité trop faible est synonyme de manque de flexibilité pour modéliser le problème. Une complexité trop importante peut aboutir

à un sur-apprentissage. Dans les deux cas, les performances de généralisation sont amoindries.

Dans le livre d'Haykin (1999) sont listées les principales règles d'apprentissage. Les règles définissent la manière dont les poids des neurones sont ajustés au cours de l'apprentissage.

L'apprentissage est aussi une façon de prendre en compte l'environnement. Nous parlons alors de paradigmes d'apprentissage. La disponibilité de l'information relative à l'environnement permet de distinguer trois paradigmes :

- L'apprentissage est *non-supervisé* ou *auto-organisé* quand l'adaptation des poids ne dépend que des critères internes au réseau. L'adaptation ne se fait qu'avec les signaux d'entrées. Aucun signal d'erreur, aucune sortie désirée n'est prise en compte. Les poids reflètent certaines caractéristiques statistiques des données d'entrées. Nous aurons l'occasion de vérifier cette propriété dans l'étude des cartes de Kohonen.
- L'apprentissage est *supervisé* quand il est possible de fournir une sortie désirée (un modèle). Quand un réseau apprend une fonction multivariable, il réalise une régression de fonction à partir d'un jeu d'entrées et de sorties correspondantes. Les poids du réseaux de neurones sont ajustés en fonction d'un signal d'erreur. Celui-ci est la différence entre la sortie fournie par le réseau et la sortie désirée (fournie par un modèle).
- L'apprentissage est par *renforcement* ou par *assignation de crédits* quand le réseau de neurones interagit avec l'environnement. Contrairement à l'apprentissage supervisé, l'environnement ne fournit pas les réponses désirées, mais assigne une récompense quand la réponse du réseau est satisfaisante, ou assigne une pénalité si la réponse du réseau n'est pas satisfaisante. Le réseau doit ainsi découvrir les réponses qui lui donnent un maximum de récompenses.

Après le cycle d'apprentissage, le réseau de neurones est capable de répondre à n'importe quelle entrée. Il est intéressant de noter que le réseau neuronal n'utilise pas de modèle prédéterminé, comme le font les approches statistiques. Il permet d'approximer avec précision n'importe quelle fonction linéaire ou non linéaire.

Le choix du réseau, et par conséquent de l'algorithme, va essentiellement dépendre de l'application et de la disponibilité des informations.

En robotique, nous utilisons principalement les apprentissages supervisés, comme par exemple Ritter *et al.* (1992), ou les apprentissages par renforcement, comme par exemple Adda *et al.* (2004).

2.2 L'utilisation du connexionnisme en robotique

C'est surtout vers la fin des années 80 que les premières applications robotiques utilisant le connexionnisme font leur apparition. Citons par exemple les travaux de Kawato *et al.* (1987), de Kuperstein (1988).

Les travaux actuels concernant le connexionnisme dans les tâches robotiques sont divers et variés. Il serait difficile d'en faire une liste exhaustive. Plusieurs ouvrages réunissent les diverses contributions dans le domaine. Citons par exemple Miller III *et al.* (1990) ; IEEE SMC (1996) ; NN (1998) ; RAS (2003). Dans ce qui va suivre, nous allons plutôt nous intéresser à la plate-forme robot-vision du laboratoire et présenter les différents travaux effectués autour de celle-ci.

Les réseaux de neurones sont principalement utilisés en robotique pour déterminer des modèles de transformations géométriques ou cinématiques. Ces modèles établissent la corrélation existante entre les entrées du système et les sorties correspondantes.

Les réseaux de neurones permettent aussi de déterminer les modèles inverses, c'est-à-dire la corrélation entre l'espace des sorties et l'espace des entrées. Il s'agit donc de déterminer le jeu d'entrées qui correspond à une sortie désirée.

Ces modèles sont difficiles à obtenir car souvent les fonctions de corrélations sont grandement non-linéaires. Les paramètres intervenant dans ces modèles peuvent ne pas être mesurables ou évoluer dans le temps par exemple. Ceci rend difficile leurs prises en compte et complexifie grandement la tâche de modélisation. Les réseaux de neurones permettent d'identifier des systèmes non-linéaires, avec de nombreuses variables. Ils s'adaptent eux-mêmes au stockage des informations nécessaires.

La plate-forme robot-vision

Les travaux du laboratoire concernent principalement le contrôle de bras manipulateur, de tête robotique et de vision. Ces travaux s'articulent autour d'une plate-forme expérimentale. Elle permet à l'équipe d'affiner et de valider des algorithmes neuromimétiques dans le contexte de l'asservissement visuel de systèmes robotisés.

Cette plate-forme est composée d'un bras manipulateur, Sylvia, accroché au plafond et à six degrés de liberté (DOF). Il est équipé de différents capteurs proprioceptifs qui nous renseignent sur l'état du système. Ces informations sont les positions articulaires, les vitesses, les couples d'effort, etc.

Pour tenir compte de l'environnement, il faut adjoindre des capteurs extéroceptifs. Pour cela, la plate-forme est équipée d'une tête robotique sur laquelle sont montées deux caméras. L'extrême richesse de l'information visuelle peut renseigner sur la position du robot, la position de cibles ou la présence de tout autre obstacle.

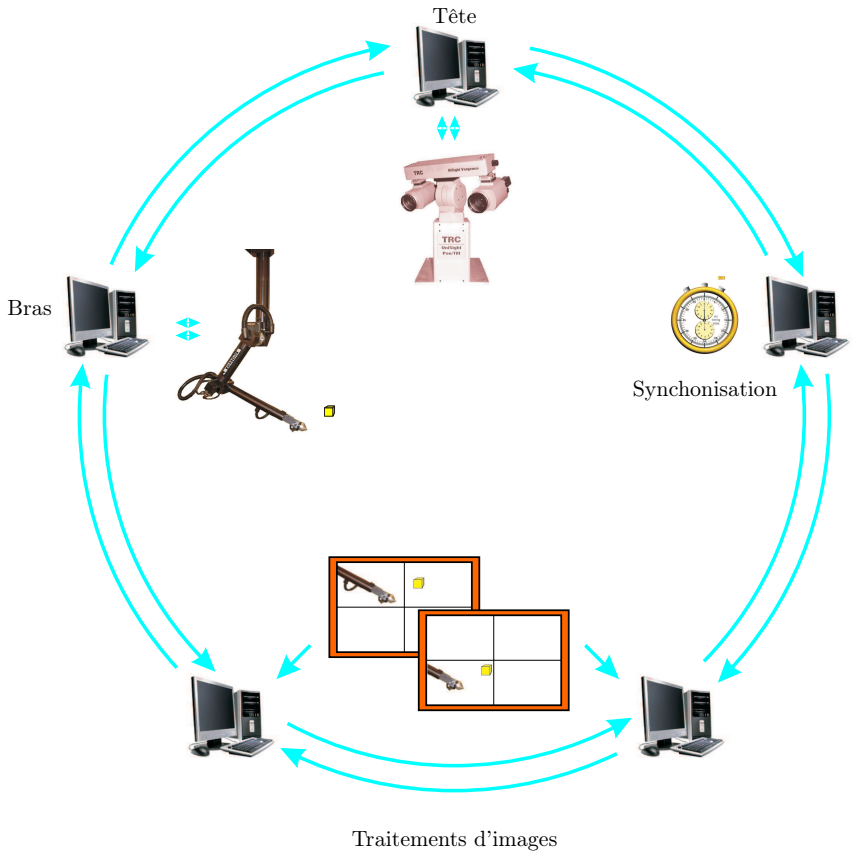


Fig. 2.3 *Plate-forme expérimentale*

Des ordinateurs de type PC fournissent la puissance de calcul nécessaire pour faire fonctionner la plate-forme. Dans sa forme actuelle, celle-ci comporte cinq ordinateurs (figure 2.3). Deux sont utilisés pour commander respectivement le bras et la tête. Deux autres sont utilisés pour les traitements d'images. Un dernier s'occupe de collecter et dater toutes les informations disponibles afin de pouvoir synchroniser toutes les données.

À l'heure actuelle, cette plate-forme est en pleine restructuration afin de pourvoir répondre aux exigences des nouvelles manipulations à venir. Ces modifications interviennent au niveau des traitements d'images (avec notamment l'ajout de la couleur (Hermann *et al.*, 2004a ; Buessler *et al.*, 2004)), de la mise en place d'un superviseur (pour la synchronisation et l'administration de la plate-forme) et d'une mise à niveau du matériel (remplacement des contrôleurs d'axes, pour répondre aux nouvelles spécifications de l'informatique).

Les contributions du laboratoire

Daniel Kuhn (1997) montre les possibilités offertes par les techniques neuromimétiques dans le domaine du contrôle robotique par boucle de retour d'informations visuelles (vision monoculaire et stéréoscopique). Quelques réalisations illustrent l'efficacité de l'approche, mais se limitent à la gestion de déplacements dans le plan avec une seule caméra.

Stéphane Breton (1999) propose une approche dédiée à la coordination sensori-motrice d'un système de vision binoculaire et d'un bras manipulateur. Les informations visuelles binoculaires sont traitées en deux modules : le premier s'occupe de la reconstruction de l'espace 3D dans lequel évolue le bras manipulateur, et le second utilise les informations du premier pour asservir le robot. Stéphane Breton utilise deux techniques statistiques du traitement du signal : l'analyse en composantes principales (ACP) et l'analyse en composantes indépendantes (ACI).

Jean-Luc Buessler (1999) explore la possibilité de créer des architectures modulaires en composant les réponses de plusieurs réseaux neuronaux. Il montre que l'apprentissage peut être réalisé en introduisant une architecture bidirectionnelle. Chaque module apprend à la fois la transformation directe et la transformation inverse entre ses entrées et ses sorties.

Wiliam Jacquet (2001) propose deux approches pour l'estimation de la fonction inverse du robot. La première est basée sur le réseau GTM (*Generative Topographic Mapping*), hors ligne à variables latentes, qu'il étend en lui adjoignant une couche de sorties, ainsi qu'un mécanisme de recrutement. La deuxième est sous forme de plusieurs réseaux de quantification vectorielle, les réseau HYPSOM. Ils adaptent automatiquement leur structure à la fonction estimée en respectant un critère d'erreur.

Les travaux de Patrice Wira (2002) s'inscrivent dans le domaine de l'utilisation de la vision pour la réalisation de tâches robotiques et concernent plus particulièrement les aspects de poursuite de cible par asservissement visuel. Il améliore la qualité de la poursuite en faisant une estimation robuste du mouvement. Cette estimation est basée sur l'aspect adaptatif du filtrage de Kalman.

Reda Kara (2003) utilise le réseau de neurones CMAC (*Cerebellar Model Articulation Controller*) pour réaliser des contrôleurs performants pour l'estimation de fonctions robotiques. Il a formalisé deux architectures neuronales modulaires, le HCMAC (*Hierarchical CMAC*) et le AL_CMAC (*Adaptive Linear CMAC*), pour résoudre les problèmes de sorties discrètes et de taille liée directement à la précision souhaitée.

2.3 Les cartes de Kohonen

α.– Le choix de cet algorithme

Les tâches robotiques qui nous intéressent peuvent être le suivi de cible, ou encore atteindre une cible quelconque de l'espace 3D, avec plus ou moins de contraintes.

L'algorithme d'apprentissage du réseau de neurones qui compose le contrôleur robotique doit avoir certaines capacités telles que l'apprentissage en ligne et une capacité d'adaptation permanente. L'adaptation permanente laisse supposer un apprentissage itératif, c'est-à-dire un apprentissage qui modifie les poids des neurones à chaque fois

qu'une nouvelle entrée est présentée. C'est un apprentissage *stochastique* puisque l'ordre de présentation des entrées est aléatoire. Il faut cependant garder à l'esprit qu'il est difficile en pratique d'obtenir des entrées complètement aléatoires, puisque celles-ci sont fournies à la cadence vidéo (dans le cas de l'apprentissage en ligne) et que par conséquent l'effecteur du robot suit une trajectoire. Pour que l'adaptation des poids puisse se faire en ligne, l'algorithme doit rester simple et ne pas être gourmand d'un point de vue ressources informatiques.

Pour ces raisons, et d'autres encore, comme la stabilité de l'apprentissage ou la rapidité de convergence de l'algorithme, nous avons choisi d'utiliser des cartes auto-organisatrices.

Les cartes auto-organisatrices de Kohonen sont relativement faciles à mettre en œuvre et bénéficient d'un apprentissage efficace dû aux principes de compétition entre les neurones et de voisinage. D'autres cartes auto-organisatrices, telle que le Neural-Gas proposé par (Martinetz et Schulten, 1991), auraient pu convenir. Les relations de voisinage de Kohonen nous semblent être plus pertinentes et plus efficace, surtout pour un apprentissage en ligne.

β.– Les algorithmes de base

Les cartes de Kohonen sont des cartes auto-organisatrices (SOM pour l'acronyme anglais *Self-Organizing Maps*) proposé par Kohonen en 1982 (Kohonen, 1982b,a, 1984).

Les cartes de Kohonen réalisent une discrétisation, une cartographie, de l'espace d'entrées. L'espace d'entrées est découpé en zones, chacune représentée par un *vecteur référent*.

Une carte de Kohonen est représentée par une grille régulière, où chaque intersection correspond à un neurone. Cette grille peut être mono- ou multidimensionnelle. Quand celles-ci sont utilisées pour l'apprentissage de fonctions, il est préférable de ne pas utiliser des cartes de dimensions supérieures à trois, car au-delà, il devient difficile à faire converger. La grille définit ainsi les relations de voisinage entre les neurones, donc les interactions des uns par rapport aux autres. Le voisinage est l'une des caractéristiques de la carte de Kohonen qui permet à l'apprentissage d'être rapide et stable. Les poids d'un neurone associé à une zone de l'espace d'entrées doivent tendre, au cours de l'apprentissage, vers les valeurs représentatives de celle-ci.

L'algorithme itératif d'apprentissage comporte deux phases distinctes (voir figure 2.4), répétées à chaque fois qu'un vecteur d'entrées $\mathbf{v}$ est présenté. Ces deux phases sont :

1. une phase de compétition entre tous les neurones de la carte ;
2. suivie d'une phase d'adaptation des poids des neurones.

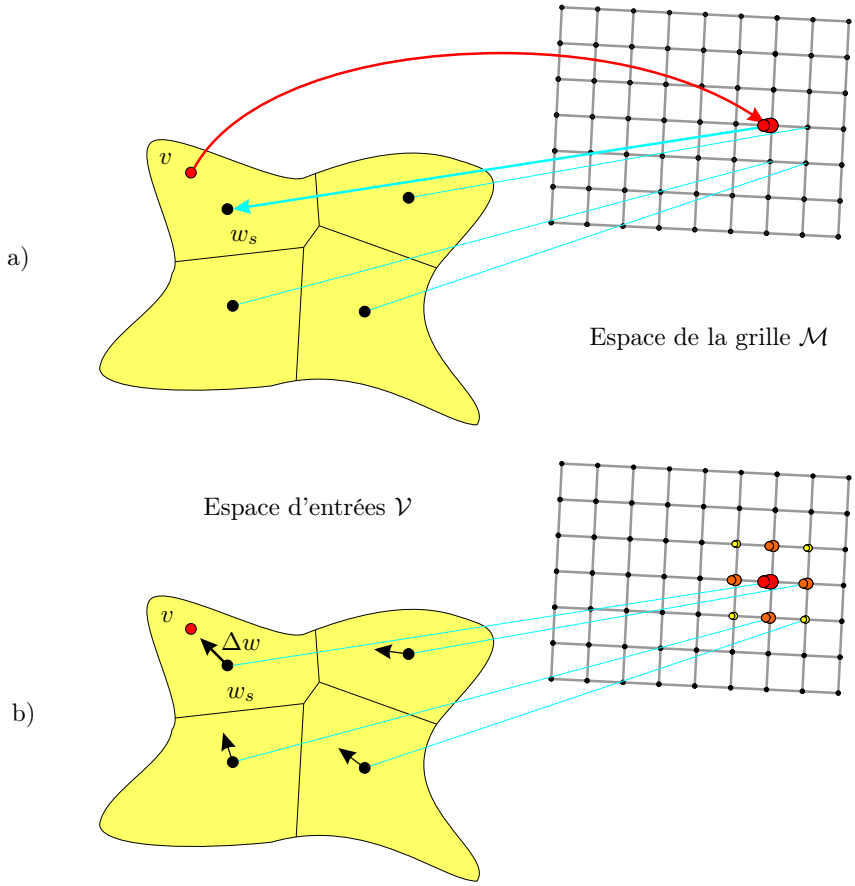


Fig. 2.4 *Principe d'une carte de Kohonen (2D)*

La vue a) représente la phase de compétition. Une entrée tirée aléatoirement sélectionne le neurone vainqueur (winner takes all). la vue b) représente la phase d'adaptation des poids. Le taux d'adaptation dépend de la distance dans la grille d'un neurone r par rapport au vainqueur s.

Soit $\mathcal{V}$ l'espace d'entrées à discrétiser, où chaque élément est représenté par un vecteur $\mathbf{v}$. Nous avons donc $\mathbf{v} \in \mathcal{V}$.

Soit $\mathcal{M}$ une grille de neurones dans laquelle les neurones sont repérés par leur position r. Le poids synaptique du neurone r est w_r.

La phase de compétition consiste en la détermination d'un neurone vainqueur s. À l'instant k, un vecteur d'entrées, $\mathbf{v}_k$, tiré aléatoirement dans l'espace $\mathcal{V}$, est présenté au réseau de neurones. Tous les neurones de la grille entrent en compétition. Le neurone vainqueur s est celui dont la distance entre le vecteur d'entrées $\mathbf{v}_k$ et son vecteur poids est la plus courte. C'est-à-dire :

$$w_s = \arg\min_{r \in \mathcal{M}} \|\mathbf{v}_k - w_r\| \quad . \tag{2.1}$$

Rappelons que l'algorithme est itératif. C'est-à-dire que les poids des neurones sont modifiés après la présentation de chaque entrée. Le neurone vainqueur est celui qui subit la plus forte modification, celui qui se rapproche le plus de l'entrée $\mathbf{v}_k$ présentée. Le neurone vainqueur provoque l'adaptation de tous les neurones de son voisi-

nage. Cette influence se représente par une fonction de voisinage h. Cette fonction h pondère la manière dont chaque neurone voisin est modifié. Cette pondération est fonction de la distance dans la grille $\mathcal{M}$ séparant un neurone du neurone vainqueur. Les poids des neurones sont alors modifiés de la manière suivante :

$$w_r = w_r + \Delta w_r \ , \tag{2.2}$$

avec :

$$\Delta w_r = \mu \cdot h_{rs} \cdot (\mathbf{v}_k - w_r) \ , \tag{2.3}$$

où μ représente le taux d'apprentissage, généralement en fonction du temps et h_{rs} le coefficient de voisinage qui est fonction de la position r d'un neurone par rapport au neurone vainqueur s.

La convergence de l'algorithme dépend fortement de la fonction de voisinage et du taux d'apprentissage. La convergence est plus rapide quand la fonction de voisinage diminue avec la distance (entre le neurone r et le neurone vainqueur s)(Ritter et Schulten, 1986), et quand le taux d'apprentissage et le rayon de voisinage diminuent pour atteindre asymptotiquement les valeurs d'équilibre de la carte (Kohonen, 1982b, 1995). Nous prenons les fonctions introduites par Ritter *et al.* (1992) (voir figure 2.5).

Le taux d'apprentissage évolue en fonction du temps de la manière suivante :

$$\mu = \mu_i \left(\frac{\mu_f}{\mu_i} \right)^{\frac{k}{k_{max}}} \ , \tag{2.4}$$

avec μ_i la valeur initiale du coefficient d'apprentissage et μ_f la valeur finale.

La fonction de voisinage évolue en fonction du temps de la manière suivante :

$$h = \exp\left(-\frac{\|r - s\|^2}{2\sigma^2} \right) \ , \tag{2.5}$$

où σ représente le rayon de voisinage. Il évolue en fonction du temps de la manière suivante :

$$\sigma = \sigma_i \left(\frac{\sigma_f}{\sigma_i} \right)^{\frac{k}{k_{max}}} \ , \tag{2.6}$$

avec σ_i la valeur initiale du rayon de voisinage et σ_f la valeur finale. L'évolution des coefficients σ et μ est montrée sur la figure 2.6.

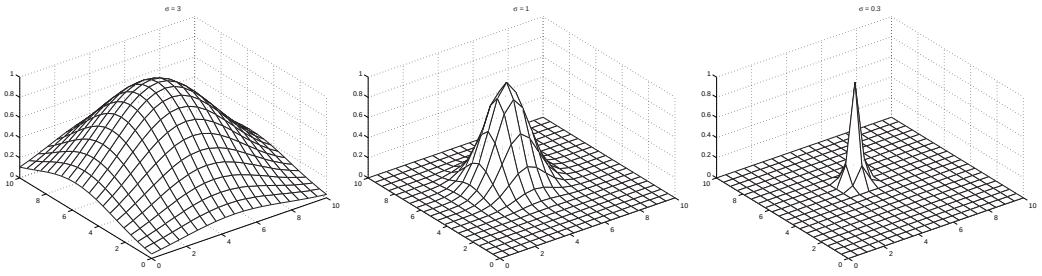


Fig. 2.5 *Évolution du rayon de voisinage*

a) b) c)

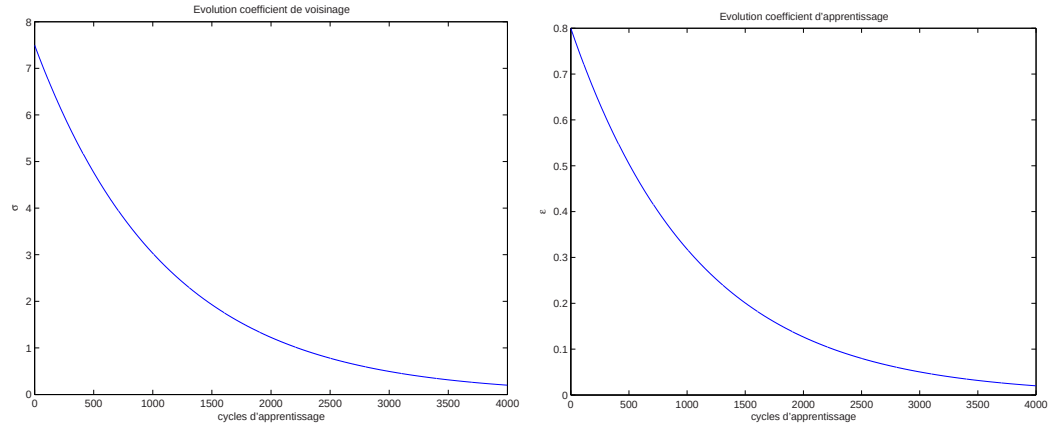


Fig. 2.6 *Évolution des coefficients σ et μ*

a) b)

La courbe a) représente l'évolution du coefficient de voisinage σ au cours du temps, la courbe b), représente quant à elle, l'évolution du coefficient d'apprentissage μ.

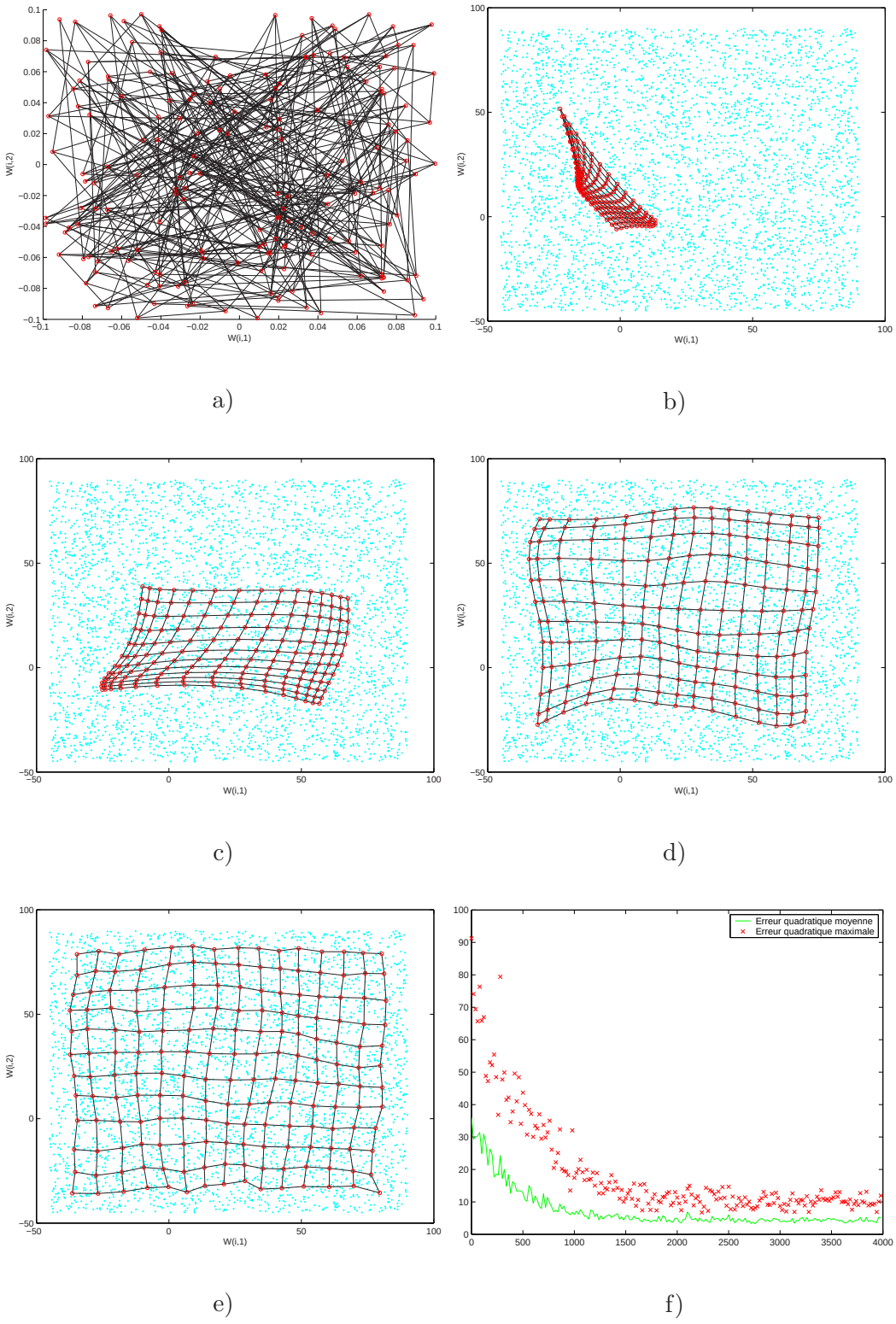
γ.– Propriétés des cartes de Kohonen

Déploiement de la carte

Il est intéressant de remarquer deux phases dans le déploiement des cartes de Kohonen. Une première phase, assez courte, où le coefficient d'apprentissage et le rayon de voisinage sont grands. Ceci permet de modifier rapidement les poids des neurones et de réaliser une discrétisation grossière de l'espace d'entrées. La puissance et l'originalité de ce genre de carte est d'utiliser une fonction de voisinage, qui permet aux neurones de passer rapidement d'un ordre tout à fait aléatoire (figure 2.7a)), à quelque chose de beaucoup plus ordonné (figure 2.7c)).

La seconde phase est quant à elle bien plus longue. C'est une phase d'affinement. Les coefficients σ et μ sont plus faibles, ce qui permet d'ajuster au mieux les valeurs finales des poids des neurones.

Il est à noter que l'apprentissage est non-supervisé. Il n'y a que les entrées du réseau qui participent à l'apprentissage et donc au déploiement de la carte. À chaque présentation d'une nouvelle entrée, les poids des neurones sont modifiés suivant deux



a) b)

c) d)

e) f)

Fig. 2.7 *Déploiement d'une carte de Kohonen*

La figure a) représente l'initialisation aléatoire des neurones dans l'espace d'entrées. Les quatre images suivantes montre l'evolution du déploiement de la carte, après 20 cycles (b)), 400 cycles (c)), 1000 cycles et 4000 cycles. Les courbes f) montrent l'évolution de l'erreur quadratique au cours du temps (moyenne et miximum).

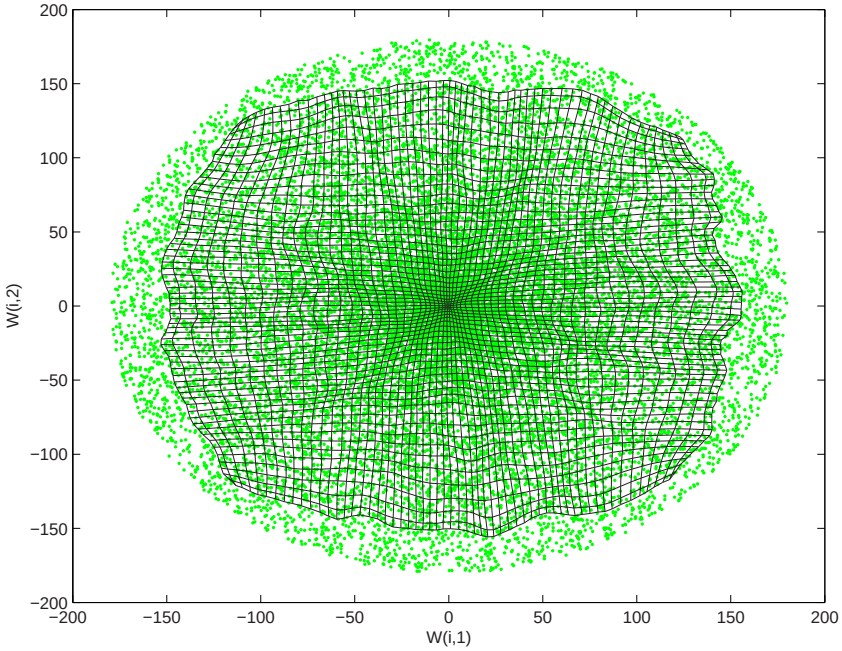


Fig. 2.8 *Exemple de discrétisation non-uniforme*

principes : la compétition, avec le choix d'un neurone vainqueur, et la coopération, avec l'influence du vainqueur sur ses voisins. La carte finale discrétise l'espace d'entrées. La distribution des neurones dans l'espace d'entrées est proche de l'optimale, même si nous pouvons observer des petites distorsions sur les bords.

La discrétisation de l'espace d'entrées tient compte de la densité des exemples présentés. Si lors de l'apprentissage, des zones de l'espace d'entrées ont une probabilité plus importante d'être présentées, alors le nombre de neurones représentatif de cette zone sera lui aussi plus important. La carte reflète la distribution des exemples tirés dans l'espace d'entrées. Le figure 2.8 montre une carte déployée dans un espace d'entrées non-uniforme.

Ordre topologique

Outre le fait de discrétiser l'espace d'entrées, les cartes de Kohonen préservent l'ordre topologique de l'espace d'entrées. Cela signifie que deux neurones voisins dans la grille sont représentatifs de deux régions voisines dans l'espace d'entrées.

Des incidents peuvent être observés lorsque l'ordre topologique n'est pas conservé. Nous pouvons alors observer des retournements ou plus simplement des cartes mal déployées. Des critères de conservation de topologie ont été proposés par Martinetz et Schulten (1994) et mis en œuvre dans un outil d'analyse proposé par Villmann *et al.* (1997). Herrmann *et al.* (1997) proposent des mesures de préservation de topologie dans des cartes entraînées avec des données réelles ou synthétiques. Aupetit (2003) propose une approche plus robuste que celle proposée par Martinetz et Schulten pour la création de graphes topographiques en considérant la densité de distribution des données. Plus récemment Hetel *et al.* (2004) ont mis au point des critères,

évalués « en ligne », qui permettent de juger de la qualité de la carte au cours du déploiement, et de l'arrêter en cas de détection de défaut.

À l'heure actuelle, la convergence des cartes de Kohonen n'a toujours pas été complètement démontrée. Le seul cas abouti est celui d'une carte mono-dimensionnelle, nous pouvons nous référer à Erwin *et al.* (1992) par exemple. Pour les autres cartes, les résultats ne sont que partiels.

Évolution des paramètres et initialisations

Il est également intéressant de faire quelques remarques importantes concernant les entrées, le réglage des paramètres ou encore sur la convergence. Ces remarques assez générales restent vraies pour les extensions que nous verrons un peu plus loin.

En règle générale, toutes les entrées du réseau de neurones ont la même importance, c'est pourquoi il est nécessaire de faire que chacune d'entre elles participe à part égale dans l'élection du neurone vainqueur. Cela est surtout visible lorsque des grandeurs de natures différentes sont utilisées (comme par exemple des angles en radians avec des distances en mm). Il faut alors conditionner les variables d'entrées, les normaliser, afin de les rendre comparables (en terme de valeurs) les unes par rapport aux autres.

Bien que les réseaux de neurones aient des capacités d'adaptation assez convainquantes, pour obtenir les meilleurs résultats possibles, il faut choisir de bons paramètres initiaux et avoir une évolution des coefficients adaptée au problème. À titre d'exemple, sans connaissances a priori de l'espace à discrétiser, les poids initiaux des neurones auront des valeurs petites et aléatoires. Les taux d'apprentissage et de voisinage devront alors être suffisamment grands durant un laps de temps assez long pour permettre à l'algorithme de déplacer les neurones dans la zone à discrétiser. Par contre, si l'espace d'entrées est connu, il est possible de répartir les neurones dans cet espace. Les taux d'apprentissage seront alors plus faibles.

Pour que les poids des neurones convergent vers des valeurs optimales, il est indispensable que le coefficient d'apprentissage devienne faible au cours de l'apprentissage. Garder des coefficients forts ne crée pas de sur-apprentissage, mais fait osciller les poids des neurones dans une zone de l'espace.

δ.– Les cartes de Kohonen linéaires locales (LLM)

Dans le cas de la robotique, les fonctions à apprendre lient un espace d'entrées à un espace de sorties. Ces fonctions sont généralement non-linéaires quelconques, et sont continues. Nous allons, pour apprendre ce genre de fonction, utiliser une carte de Kohonen étendue à sorties linéarisées. Une couche de sorties est associée à la carte SOM de base.

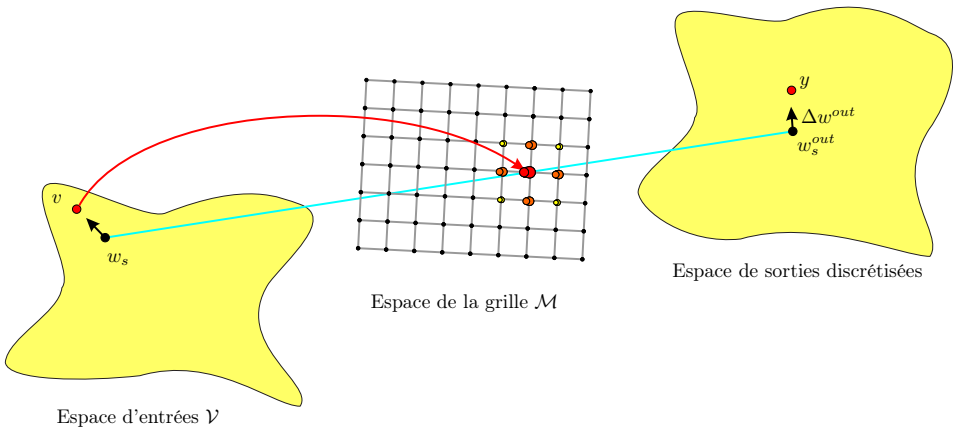


Fig. 2.9 *Carte auto-organisatrice étendue*

Les cartes SOM–LLM (*Linear Local Map*) sont le fruit de la combinaison d'une carte de Kohonen étendue, qui fournit une sortie discrète, associée à des modèles linéaires : les ADALINES (*ADAptive LINear Element*)

La carte de Kohonen étendue

Ritter et Schulten (1986) ont proposé d'adjoindre une couche de sorties au modèle de carte initialement proposé par Kohonen (voir figure 2.9). La couche de sorties permet d'utiliser les cartes auto-organisatrices pour des applications robotiques (ou toutes autres régressions) tout en gardant les propriétés avantageuses citées plus tôt.

Dans ces cartes, un second vecteur de poids synaptiques est associé à chaque neurone. Nous appellerons ces coefficients *poids de sorties* (w_r^{out} est le poids associé au neurone r). La couche d'entrées n'est pas modifiée par cette extension. La sélection du neurone vainqueur s se fait donc toujours dans l'espace d'entrées selon l'équation 2.1. La sortie $\mathbf{y}$, réponse à une entrée $\mathbf{v}$ est :

$$\mathbf{y} = w_s^{out} \quad . \tag{2.7}$$

Un neurone de la carte d'entrées représente toute une zone de l'espace d'entrées. Par conséquent la réponse sera la même pour toute la zone. Cette nouvelle carte de Kohonen fonctionne comme une table associative. Une entrée sélectionne un neurone vainqueur qui fournit la sortie associée.

L'apprentissage des poids de sorties est supervisé. Il faut donc présenter en même temps que l'entrée de sélection $\mathbf{v}$, la sortie désirée $\mathbf{y}$. Les poids de sorties sont adaptés de la même manière que les poids d'entrées.

$$w_r^{out} = w_r^{out} + \Delta w_r^{out} \quad , \tag{2.8}$$

avec :

$$\Delta w_r^{out} = \mu' \cdot h'_{rs} \cdot (\mathbf{y}_k - w_r^{out}) \ . \tag{2.9}$$

Les coefficients μ' et σ' sont respectivement le taux d'apprentissage et le rayon de voisinage pour la couche de sorties.

Les SOM à réponses linéaires

Les techniques linéaires locales sont très attractives pour la modélisation de séries temporelles complexes (Principe *et al.*, 1998). Elles sont aussi utilisées de plus en plus souvent pour la commande de systèmes.

L'ADALINE est un modèle linéaire adaptatif introduit par Widrow et Hoff en 1960. Une synthèse complète sur ces réseaux se trouve dans Widrow et Walach (1996). Les ADALINES retournent une sortie $\mathbf{y}$, somme pondérée d'un vecteur d'entrées $\mathbf{u}$ par une matrice $\mathbf{A}$. Cette matrice est adaptée au cours de l'apprentissage.

$$\hat{\mathbf{y}} = \mathbf{A} \cdot \mathbf{u} \ . \tag{2.10}$$

Les poids de l'ADALINE sont adaptés par l'algorithme LMS normalisé (*Least Mean Square*) (Widrow et Lehr, 1990).

$$\mathbf{A}_{k+1} = \mathbf{A}_k + (\mathbf{A}_k^* - \mathbf{A}_k) \ , \tag{2.11}$$

où $\mathbf{A}_k^*$ est un modèle de poids avec :

$$\mathbf{A}_k^* = \mathbf{A}_k + \frac{\epsilon_k \cdot \mathbf{u}_k}{\|\mathbf{u}_k\|^2} \quad \text{avec} \quad \epsilon_k = \mathbf{y} - \hat{\mathbf{y}} \ . \tag{2.12}$$

Ritter et Schulten (1988) proposent une seconde extension (le SOM–L). Ils associent à chaque neurone de la carte de Kohonen un ADALINE. La sélection du neurone vainqueur se fait dans l'espace d'entrées et la sortie est fournie par l'ADALINE associé à ce neurone. Deux vecteurs doivent être présentés à ce réseau. Un premier, $\mathbf{v}$, sert à la détermination du vainqueur s, et un second, $\mathbf{u}$, sert d'entrée pour l'ADALINE. Seul l'ADALINE $\mathbf{A}_s$ participe au calcul de la réponse. En revanche, pour accélérer le processus d'apprentissage et profiter des capacités des cartes auto-organisées, non seulement les poids de l'ADALINE vainqueur seront modifiés mais aussi les poids de ses voisins. L'équation 2.11 devient alors :

$$\mathbf{A}_{r,k+1} = \mathbf{A}_{r,k} + \mu' \cdot h'_{rs}(\mathbf{A}^*_{s,k} - \mathbf{A}_{r,k}) \ . \tag{2.13}$$

Les deux approches que nous venons de voir peuvent se combiner. Nous obtenons ainsi un SOM–LLM (voir figure 2.10). Une couche de sorties, qui donne une réponse

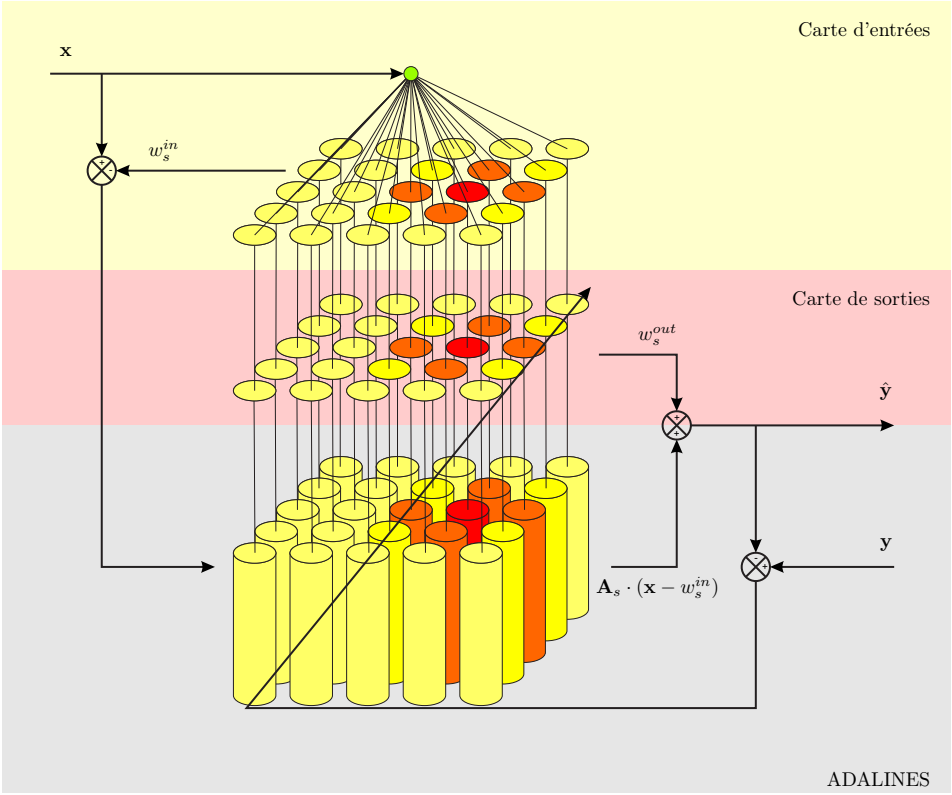


Fig. 2.10
Le SOM–LLM

discrète, est associée à la carte de Kohonen de base, et des ADALINES, qui affinent la réponse discrète, sont associés à la couche de sorties. Ce réseau fournit une réponse affine à une entrée $\mathbf{x}$.

La sélection du neurone vainqueur s se fait dans la couche d'entrées. La sortie du réseau est alors :

$$\hat{\mathbf{y}} = w_s^{out} + \mathbf{A}_s \cdot (\mathbf{x} - w_s^{in}) \tag{2.14}$$

Pour l'apprentissage supervisé, il faut définir des modèles pour les adaptations des poids de la carte de sorties et pour les ADALINES. Les modèles sont symbolisés par une étoile en exposant.

$$w^* = \mathbf{y} - \mathbf{A}_s \cdot (\mathbf{x} - x^{in_s}) \ , \tag{2.15}$$

$$\mathbf{A}^* = \mathbf{A}_s + \frac{(\mathbf{y} - \hat{\mathbf{y}}) \cdot (\mathbf{x} - w_s^{in})^T}{\|\mathbf{x} - w_s^{in}\|^2} \ . \tag{2.16}$$

Les poids sont ensuite modifiés en fonction des taux d'apprentissage et des rayons de voisinage :

$$w_{r,k+1}^{in} = w_{r,k}^{in} + \mu \cdot h_{rs} \cdot (\mathbf{x}_k - w_{r,k}^{in}) \ , \tag{2.17}$$

$$w_{r,k+1}^{out} = w_{r,k}^{out} + \mu' \cdot h'_{rs} \cdot (w^* - w_{r,k}^{out}) \ , \tag{2.18}$$

$$\mathbf{A}_{r,k+1} = \mathbf{A}_{r,k} + \mu'' \cdot h''_{rs} \cdot (\mathbf{A}^* - \mathbf{A}_{r,k}) \ . \tag{2.19}$$

Cette dernière variante de carte est celle que nous allons utiliser pour nos contrôleurs robotiques puisqu'elle allie capacité d'apprentissage de fonctions non-linéaires en ligne, rapidité de convergence, facilité de mise en œuvre,... Le fait de passer d'une réponse discrète à une réponse linéarisée diminue fortement l'erreur d'estimation pour un nombre donné de neurones. Il faut cependant rester attentif au fait que le réseau doit comporter assez de neurones pour être représentatif de la fonction à apprendre. Elle doit être topologiquement lisse (Buessler, 1999).

ε.– D'autres réseaux à base de carte auto-organisatrices

De nombreuses autres variantes ou extensions des cartes SOM ont été proposées. Le livre de Kohonen (1995) en donne un bon aperçu. Le numéro spécial du Neural Networks NN (2002) présente des développements plus récents. Nous n'allons pas les reprendre ici, nous allons simplement en présenter d'autres, intéressants, d'un point de vu de l'architecture ou encore du point de vu caractéristiques de neurones.

Intégration de la notion de temps

Stringer *et al.* (2003, 2004) proposent une architecture neuronale à attracteurs continus. Cette forme d'apprentissage temporel est capable d'apprendre des séquences temporelles de comportement. Ils utilisent cette approche pour apprendre et exécuter des séquences de mouvements.

L'idée d'introduire un facteur temps n'est pas nouvelle. Cette dimension temporelle peut prendre différentes formes, telle qu'une couche dédiée à la capture d'informations de vitesse des entrées, dans une structure hiérarchique de carte, ou alors telle que l'intégration de mémoires destinées à stocker l'information temporelle (Euliano et Principe, 1996).

Chappell et Taylor (1993) proposent de modifier les poids des neurones des cartes de Kohonen pour permettre au réseau d'intégrer la dimension temporelle.

Pour représenter le temps dans les réseaux de neurones, Voegtlin (2002) propose de combiner des cartes SOM à des boucles de retours. Il obtient ainsi des cartes auto-organisatrices récursives. Ces cartes sont utilisées dans des tâches de classification.

Des cartes évolutives

Les cartes auto-organisatrices ont la particularité de pouvoir s'adapter à tous types d'espaces, connus ou non. La taille des réseaux de neurones dépend essentiellement de la précision de discrétisation recherchée. Cette taille peut être délicate à prévoir a priori sans connaissances préalables sur l'espace à discrétiser. Choi et Park (1994) proposent une carte auto-organisatrice à architecture évolutive pour palier à la nécessaire prédétermination de la taille de la carte.

Les *Adaptive-Subspace SOM*

Ces cartes ASSOM (*Adaptive-Subspace SOM*) sont présentées dans Kohonen (1995) comme des détecteurs d'invariances. L'approche modulaire de ce réseau permet à l'aide d'un module par région, de détecter les points d'une région de l'espace d'entrées qui sont sujets aux mêmes transformations simples. Les propriétés de ces cartes ont été approfondies dans Kohonen (1996) ; Kohonen *et al.* (1997).

López-Rubio *et al.* (2004) proposent un nouveau modèle neuronal de cartes auto-organisatrices qui utilisent l'analyse en composantes principales. Cela revient en fait à simplifier les équations d'apprentissage des réseaux ASSOM. Ils montrent en outre que ce nouveau modèle a de meilleures performances que le réseau ASSOM.

2.4 La modularité

Les réseaux de neurones permettent de mimer efficacement des comportements de systèmes qui ne dépendent que de quelques variables. Les cartes de Kohonen, par exemple, ne sont performantes que pour une dimentionnalité inférieure ou égale à trois, quand elles sont utilisées pour l'approximation de fonctions dans un espace continu et dense. De plus, l'augmentation de la complexité d'une tâche rime bien souvent avec augmentation du nombre de neurones composant le réseau. L'effet pervers d'une solution qui consiste à augmenter significativement le nombre de poids de la structure conduit à des inconvénients tels que l'augmentation du coût algorithmique, une convergence plus lente et délicate de l'apprentissage ou encore un temps de réponse plus long.

Des améliorations peuvent être apportées à différents niveaux. Une première manière se situe au niveau du matériel. Corne (1999) propose, par exemple, une architecture parallèle pour le calcul des poids des neurones. Une seconde solution est de développer des neurones plus complexes, dans lesquels il est possible d'inclure des connaissances supplémentaires. Ces neurones se veulent être plus proches du neurone biologique. Des réseaux composés de neurones complexes ont été présentés plus haut, dans le cadre des extensions des réseaux SOM. Une troisième amélioration peut être de faire appel à la modularité. La modularité est un concept complexe, défini de diverses manières. Des classifications des approches modulaires sont proposées dans Ronco et Gawthrop (1995), Caelli *et al.* (1999) ou dans Buessler (1999) .

Nous retiendrons deux formes de modularité dans la conception d'architectures neuronales à plusieurs réseaux (Sharkey, 1996, 1997). La première est composée de plusieurs modules, chacun estimant une même grandeur. Toutes les sorties sont combinées pour fournir une réponse globale. Nous parlons alors de modularité à modèle multiple (*ensemble-based approach*). Le pendant de cette approche est la modula-

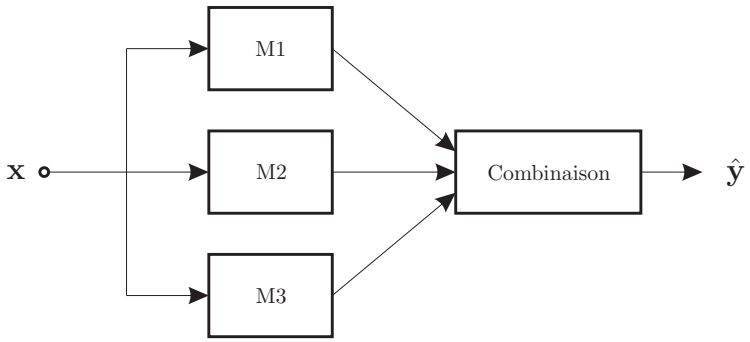


Fig. 2.11 *Modularité à modèles multiples*

rité vraie (*neural nets ensemble*). Ici, chaque réseau de l'architecture ne cherche pas à résoudre le problème dans sa globalité, mais plutôt à en résoudre un aspect. La tâche de départ est scindée en sous-tâches, plus simples à résoudre. La dépendance des modules de l'architecture peut être série ou parallèle, comme nous allons le voir ci-après.

α.– La modularité à modèles multiples

Dans le cas de la modularité à modèles multiples, tous les modules sont en parallèle et prennent le même vecteur d'entrées. L'ensemble des réseaux de neurones utilisés apprend à résoudre la même tâche. L'apprentissage supervisé peut spécialiser chacun des modules dans un secteur de l'espace des réponses. La réponse finale de toute l'architecture est déterminée, soit par le vote (ou sélection exclusive *winner takes all*), soit par la combinaison des réponses de chacun des modules. La figure 2.11 donne le schéma de principe de cette architecture.

L'avantage certain d'utiliser une telle architecture, est un risque moindre d'obtenir une sortie erronée, puisque celle-ci est estimée par plusieurs réseaux. Pour cela, il est possible de prendre des réseaux différents les uns par rapport au autres. Ces différences peuvent intervenir au niveau de la taille des réseaux, des initialisations, des paramètres d'apprentissages, etc. L'architecture est dite homogène si les réseaux sont de même natures, ou hétérogène si les algorithmes neuronaux sont différents.

Dans le cas du contrôle robotique, la sélection peut se faire en fonction des performances des modules composant l'architecture, en utilisant les erreurs d'estimations. Par exemple, Wolpert et Kawato (1998) utilisent une architecture composée de plusieurs paires de modèle inverse et modèle direct. Le premier fait une prédiction et le second détermine la commande. C'est l'erreur de prédiction qui détermine la participation d'une paire à la sortie globale.

Le mélange d'experts (*Mixture of Experts*) proposé par Jacobs *et al.* (1991) est une architecture proche de celle présentée plus haut. Elle est composée de deux types de réseaux neuronaux : plusieurs réseaux experts et un réseau superviseur (voir figure 2.12). Le superviseur détermine les coefficients de pondération de la participation de chacun des experts en fonction de l'entrée. Au cours de l'apprentissage les différents

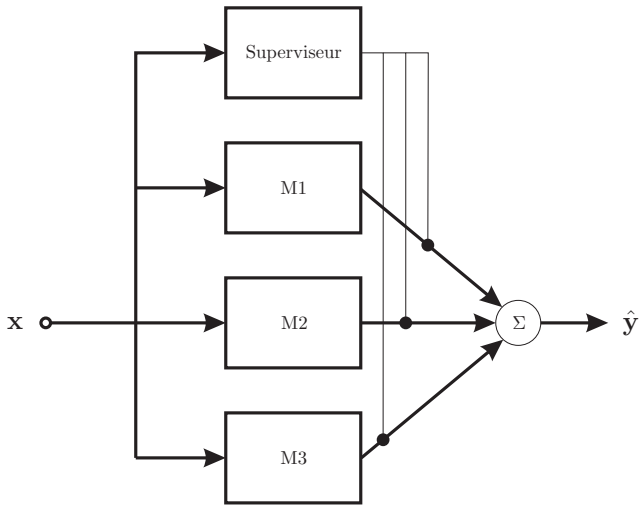


Fig. 2.12 *Mélange d'Experts*

experts se spécialisent pour un groupe de réponses parmi l'ensemble des réponses possibles. Une utilisation des mélanges d'experts dans le cadre de la robotique peut être vue dans Wira (2002). Les auteurs Jacobs et Jordan (1993) ont proposé par ailleurs un mélange d'experts hiérarchique (c'est une structure où chaque expert est en fait un mélange d'experts) pour l'identification d'un système non-linéaire. Ils montrent que ce système converge deux fois plus vite que la règle *back propagation* d'un réseau multicouche de taille comparable. Nous avons choisi de classer les mélanges d'experts parmi les architectures à modèles multiples puisque plusieurs réseaux sont mis en compétition, suite à la présentation d'un vecteur d'entrées. D'autres y verront plutôt un cas de modularité vraie, du fait qu'une tâche complexe est sous-divisée : les experts sont spécialisés pour des portions du problème.

β.– La modularité vraie

Le terme de modularité vraie reprend le principe de décomposition d'une tâche en plusieurs sous-tâches. Une telle décomposition permet de travailler avec des réseaux de petite taille. Les objectifs d'une telle architecture sont aussi bien de permettre l'apprentissage de problèmes trop complexes, qu'améliorer les performances des réseaux (temps de traitement, coût algorithmique, etc.).

Nous distinguerons deux manières d'inter-connecter les modules entre eux. Elles peuvent être parallèles ou séries. Nous parlerons d'architecture parallèle lorsque chacun des modules est indépendant et d'architecture série lorsque qu'il existe une dimension temporelle entre les modules.

Les architectures parallèles

Dans les architectures parallèles, tous les modules la composant travaillent et traitent leurs informations simultanément, comme le montre la figure 2.13.

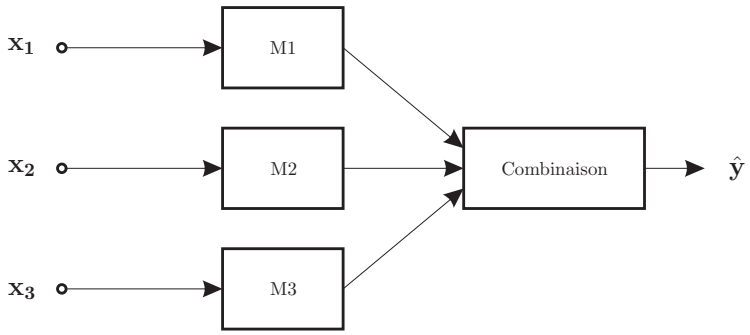


Fig. 2.13 *Architecture parallèle*

La sortie globale fait intervenir ou non chacun des modules, tout dépend des applications. La relation de coopération entre les modules peut être de type ET ou de type OU. Dans le premier cas, chacun des modules traite une partie du vecteur d'entrées, et dans le second cas, une partie seulement du vecteur d'entrées est utilisé pour le calcul de la sortie globale. La vocation première de ces combinaisons de réseaux est aussi bien de diviser une tâche en sous-tâches que d'utiliser plusieurs réseaux pour améliorer les performances en terme de précision et de fiabilité.

Cette architecture est principalement utilisée dans les applications de classification. L'architecture parallèle est aussi utilisée pour la fusion de données. Dans ce cas, l'idée est d'augmenter la fiabilité de l'information en se basant sur la redondance. En effet, les signaux d'entrées de cette architecture sont redondants, voir partiellement redondants et proviennent de divers capteurs. Les différents modules transforment ces informations en une représentation commune.

Les architectures séquentielles

Dans une architecture séquentielle, une tâche est décomposée en sous-tâches successives. L'entrée est transformée au fur et à mesure qu'elle traverse les différents modules qui composent l'architecture. Une architecture séquentielle est présentée figure 2.14. Les sorties du i-ème servent d'entrées au module i-ème+1.

Les architectures séquentielles peuvent être structurées de trois manières ;

Les structures récurrentes : dans les structures récurrentes, un ou plusieurs modules sont évalués de manière itérative. Le modèle F.I.R.M. (*forward-inverse relaxation neural networks model*),par exemple, proposé par Wada et Kawato (1993) arrive en quelques itérations à générer des trajectoires optimales.

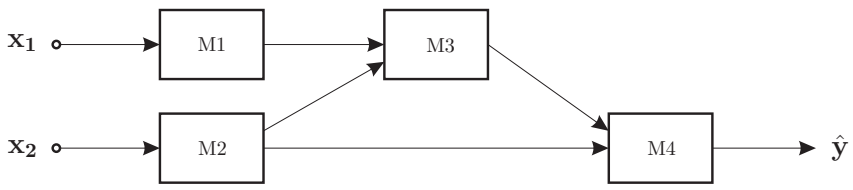


Fig. 2.14 *Architecture séquentielle*

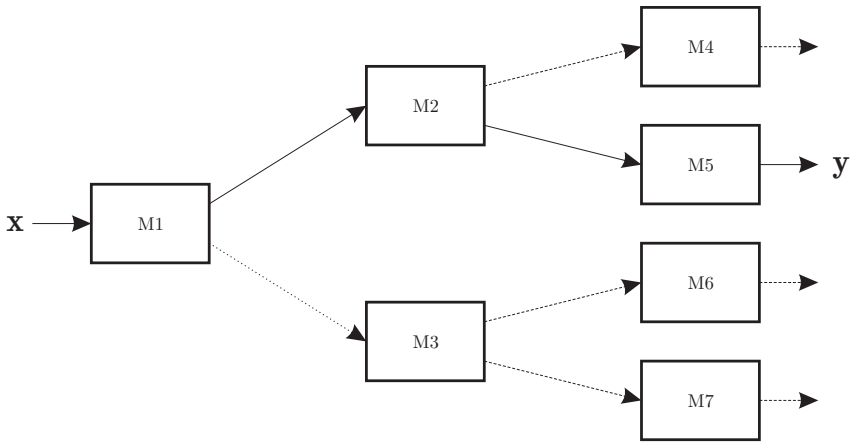


Fig. 2.15 *Architecture hiérarchique*

Les structures de composition : ce sont les structures les plus courantes. Elles ne présentent pas de cycles. Chaque module neuronal de l'architecture représente une fonction au sens mathématique du terme. À une entrée correspond une sortie. En théorie, il n'y a donc aucune difficulté à chaîner les modules et de composer, par conséquent, des fonctions complexes. Ces architectures sont très intéressantes parce qu'elles permettent de scinder l'utilisation des variables d'entées et par conséquent réduire la dimensionnalité des réseaux de neurones. En pratique, il reste difficile d'entraîner les modules. Nous reviendrons sur cet aspect un peu plus loin dans le chapitre.

Les structures convolutionnelles : elles représentent des séquences dans lesquelles un même réseau de neurones contribue à apprendre tour à tour des sous-ensembles de flux de données. Ce type de structures est spécifique à des applications qui traitent un grand nombre de variables organisées, comme les images par exemple.

Les architectures hiérarchiques

L'architecture hiérarchique est une structure en arbre. Les modules d'un niveau sont connectés à un nombre fini de modules du niveau inférieur. Un exemple d'une telle architecture est montré sur la figure 2.15. Cette architecture combine des modules en séries, et des modules en parallèles. Les modules au sein d'un niveau sont parallèles, alors que les modules qui lient une entrée à une sortie sont séries.

L'utilisation de l'architecture hiérarchique avec des SOM, pour l'approximation de fonction, a un intérêt dans la sélection du neurone vainqueur (Kohonen, 1995 ; Ritter *et al.*, 1992).

L'apprentissage d'une telle architecture ne pose aucun problème, si les différents modules sont des réseaux non-supervisés. Dans le cas de réseaux supervisés, l'apprentissage reste difficile à définir. L'apprentissage ne présente pas de problème non plus si les réseaux supervisés se situent uniquement sur la dernière couche de l'architecture. Pour illustrer ce cas, imaginons l'emploi de cette architecture avec des

SOM–LLM, pour une approximation de fonction : seule la dernière couche serait composée de tels réseaux, les précédentes quant à elles pourraient être de simples SOM qui ne serviraient, qu'à une sélection plus rapide du neurone vainqueur.

Ces architectures trouvent leur place aussi bien dans l'estimation de fonction, que dans la classification.

γ.– L'apprentissage modulaire

Nous avons vu jusqu'à présent que les réseaux neuronaux ont la capacité d'apprendre, de mimer, des systèmes non linéaires complexes. Nous avons, par ailleurs, aussi vu que ces réseaux pouvaient souffrir d'une dimensionnalité trop grande du vecteur d'entrées. La décomposition semble être une solution dans la mesure où il est possible de définir des sous-tâches n'employant que quelques variables du vecteur d'entrées. Les modules sont alors organisés selon le schéma d'architecture séquentielle défini plus haut.

Bien que les règles d'adaptation propres à chacun des modules ne changent pas, l'organisation des réseaux de neurones dans une architecture modulaire va quant à elle, rendre l'apprentissage plus délicat. Les modules d'une architecture modulaire peuvent suivre trois processus d'apprentissage différents.

L'apprentissage indépendant : entraîner les modules de manière indépendante semble être la manière la plus simple. Cela laisse supposer que les autres modules de l'architecture ne participent pas à l'apprentissage. L'interaction entre les modules n'intervient alors que lors de l'utilisation du système, dans la phase de restitution. Cette technique suppose que toutes les grandeurs sont accessibles, pour fournir des sorties désirées à chaque module dans le cas d'un apprentissage supervisé.

L'apprentissage séquentiel : l'apprentissage des modules de l'architecture peut être séquentiel. Les différents modules sont entraînés au fur et à mesure, en fonction de leur position. Après que le premier module a appris, ses sorties peuvent être utilisées pour entraîner les modules suivants (ou d'un niveau inférieur). Murino (1998) entraîne ces réseaux indépendamment. Il rajoute des modules à mesure que ceux des couches supérieures ont appris.

L'apprentissage coopératif : dans le cas de nombreuses applications, les seules grandeurs accessibles sont celles de l'espace d'entrées et celles de l'espace de sorties. L'idée est d'utiliser des méthodes globales qui permettent d'entraîner tous les modules en même temps. Il est nécessaire alors d'avoir une architecture figée, déterminée à l'avance.

Avec nos applications robotiques, nous nous trouvons typiquement dans la nécessité d'un apprentissage modulaire global. La figure 2.16 illustre ce que pourrait être une architecture modulaire pour nos applications.

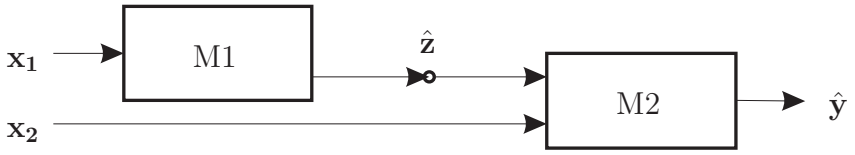


Fig. 2.16 *Exemple d'architecture modulaire*

Dans cet exemple, il y a deux modules. Chacun d'eux prend une partie du vecteur d'entrées. Il est cependant nécessaire de préciser que les composants de $\mathbf{y}$ ne peuvent pas être déterminés comme des fonctions indépendantes $f_1(\mathbf{x}_1)$ et $f_2(\mathbf{x}_2)$, mais peuvent être exprimés comme $f_2(f_1(\mathbf{x}_1),\mathbf{x}_2)$. La dimensionnalité de chacun des modules est alors plus petite que la dimensionnalité du problème de départ. Ceci à pour but, de faciliter l'apprentissage et la convergence des différents modules. Si les modules M1 et M2 ont besoin d'un apprentissage supervisé, c'est-à-dire de sorties désirées, nous voyons bien que pour entraîner M1 nous avons besoin de la grandeur intermédiaire z (et de y pour entraîner M2). En l'absence de connaissances sur cette variable, il faut mettre en œuvre des « structures d'aides à l'apprentissage » (Buessler, 1999). L'apprentissage bidirectionnel est une solution originale qui permet l'apprentissage d'architectures neuronales complexes avec une décomposition séquentielle des traitements.

La décomposition modulaire est la solution pour traiter des problèmes complexes. Une décomposition séquentielle permet quant à elle de scinder le traitement d'une tâche en réduisant sa dimensionnalité. L'apprentissage de cette structure doit être global pour pouvoir faire l'apprentissage en ligne. Plus la tâche va être décomposée en sous-tâches, plus il y aura de variables intermédiaires à estimer. Les modules devront rester simples et d'être pertinents. Il faut se garder de décomposer à outrance, afin d'éviter d'introduire d'inutiles variables intermédiaires, qui compliqueraient la phase d'apprentissage.

δ.– L'apprentissage modulaire bidirectionnel

Nous allons voir, dans cette section, comment entraîner les modules qui composent une architecture modulaire séquentielle.

La dimensionnalité

Les réseaux de neurones savent très bien approximer des fonctions sur des espaces continus et multidimensionnels. Cependant, les performances de ceux-ci se trouvent rapidement amoindries dans la résolution de problèmes à dimensions élevées. Ici, la dimensionnalité correspond à la taille du vecteur d'entrées. Les tâches d'asservissements visuels en robotique sont des problèmes à dimentionnalité élevée et sont, par conséquent, difficiles à mimer avec un réseau de neurones unique. La solution, pour continuer à bénéficier du mécanisme neuromimétique, est d'utiliser des architectures modulaires. La décomposition séquentielle, définie dans les sections précédentes, per-

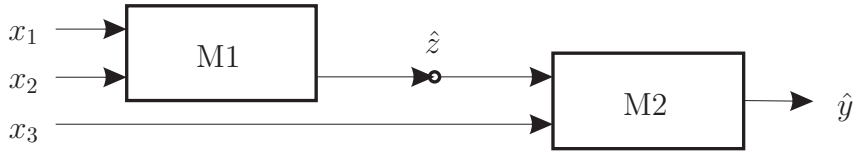


Fig. 2.17 *Exemple d'architecture modulaire*

met de résoudre des systèmes complexes (au sens dimensionnalité) du fait qu'elle permet de scinder le vecteur d'entrées et donc de réduire la dimentionnalité d'une sous-tâche.

Une décomposition séquentielle nécessite une étude préalable du problème, qui permet de dégager une composition de n fonctions $f_n()$ équivalentes à une fonction $F()$. Il faut que chacune des sous-fonctions f_n soit de dimentionnalité inférieure à celle de $F()$. Pour que la décomposition puisse se faire, il faut que les variables d'entrées soient suffisamment indépendantes.

La décomposition d'un système totalement inconnu n'est pas toujours possible. Cependant, les connaissances a priori que nous avons sur les systèmes physiques que nous cherchons à modéliser permettent de dégager une décomposition. L'étude préliminaire consiste en la détermination des liens entres les différentes entrées, ou encore en la détermination du rôle de chacun des modules (en s'appuyant, par exemple, sur des considérations géométriques). Elle ne consiste en aucun cas en la détermination des modèles des sous-fonctions.

La décomposition séquentielle

Typiquement, apprendre la relation entre un espace d'entrées et un espace de sorties, revient à identifier une fonction $F()$. Prenons, à titre d'exemple, une fonction $F()$ à apprendre, qui fournit une sortie y qui dépend de trois entrées x_1, x_2 et x_3.

$$y = F(x_1,x_2,x_3) \ . \tag{2.20}$$

Définir une décomposition séquentielle revient à définir deux fonctions, $f_1()$ et $f_2()$, telles que :

$$F\left(x_1,x_2,x_3\right) = f_2\left(x_3,f_1\left(x_1,x_2\right)\right) \ . \tag{2.21}$$

La figure 2.17 montre un exemple de décomposition modulaire. Sur cette figure, z est la sortie du module M1 (qui apprend $f1()$). Elle est utilisée comme entrée, avec x_3, pour le module M2 (qui apprend $f_2()$). Cette variable intermédiaire z peut être introduite dans l'équation (2.21), ce qui revient à écrire :

$$F\left(x_1,x_2,x_3\right) = f_2\left(x_3,z\right) \ , \tag{2.22}$$

avec : $z = f_1(x_1,x_2)$.

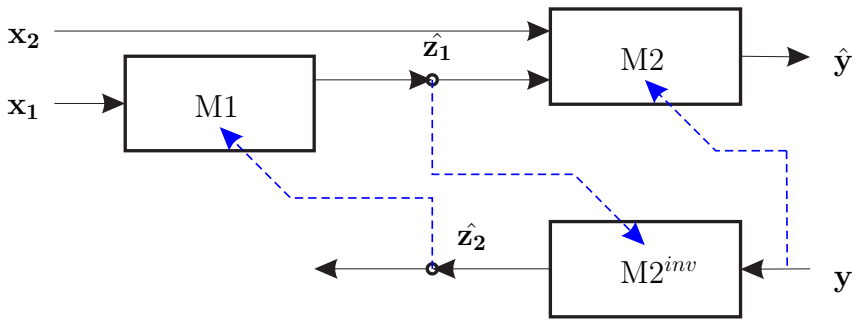


Fig. 2.18 *Exemple d'architecture bidirectionnelle*

L'apprentissage de la fonction $F()$ est supervisé. Le principe de l'apprentissage consiste en la minimisation d'un critère d'erreur. L'erreur utilisée est l'erreur d'estimation entre les sorties désirées et les sorties calculées. L'apprentissage se fait donc en utilisant le couple entrées de sélection / sorties désirées, c'est-à-dire en fonction de (x_1,x_2,x_3,y). Dans le cas de la décomposition, l'apprentissage dépend aussi de la grandeur z. Cette grandeur n'étant généralement ni connue, ni mesurable, elle doit, elle-aussi, faire l'objet d'une estimation. Cette estimation est introduite par un flux inverse, par opposition au flux direct défini par la transformation des entrées en sorties, en passant tour à tour par les modules Mn (Buessler, 1999).

Les flux bidirectionnels

Un module Mi apprend la relation entre un espace d'entrées et un espace de sorties. C'est le flux direct. La grandeur z (voir figure 2.17) est estimée par le module M1. Le module M2 prend cette estimation en entrée et fournit une réponse y. En inversant ces flux, nous obtenons un module M2^{inv}, qui fait lui aussi une estimation de z, en prenant ses entrées dans l'espace de sorties. Il faut, bien sûr, qu'il existe une fonction $g()$ telle que $\hat{\mathbf{z_2}} = g(\mathbf{y})$. Cette seconde estimation de z rend possible l'apprentissage du premier module, puisque qu'elle va servir de modèle.

L'association des modèles directs et inverses constituent des flux de données bidirectionnels. La figure 2.18 montre une structure bidirectionnelle.

L'adaptation du module M2 reste inchangé. Les entrées de sélections sont $\mathbf{x_2}$ et $\hat{\mathbf{z_1}}$, les sorties désirées sont $\mathbf{y}$. A priori, le module inverse M2^{inv} est tout aussi difficile à entraîner que le module M1, puisque ces deux modules nécessitent le modèle $\mathbf{z}$ pour apprendre. L'apprentissage de ces deux modules se fait simultanément. La sortie $\hat{\mathbf{z_1}}$ sert de modèle d'apprentissage au module M2^{inv}, alors que la sortie $\hat{\mathbf{z_2}}$ sert de modèle d'apprentissage au module M1. Ces deux représentations internes de la même grandeur doivent tendre vers la même valeur au cours de l'apprentissage.

La représentation interne $\mathbf{z}$ n'est pas unique. Elle doit être solution des deux fonctions $f_1()$ et $f_2()$. C'est-à-dire que si la variable $\mathbf{z}$ est sensée représenter une grandeur physique, elle peut l'être à un facteur d'échelle près, à un décalage près, etc.

Les règles d'adaptation

Les règles d'adaptation que nous présentons ici sont celles que nous allons utiliser. Elles concernent plus particulièrement les règles pour adapter des cartes de type SOM–LLM présentes dans l'architecture modulaire. Les poids des cartes de sorties de ces réseaux sont adaptés selon l'équation (2.18). Les modèles d'adaptation des neurones élus sont respectivement w^*_{s1} et x^*_{s2} pour les modules M1 et M2^{inv}.

Sans précaution particulière, les variables estimées $\hat{\mathbf{z_1}}$ et $\hat{\mathbf{z_2}}$ convergeraient rapidement vers une solution triviale constante, non représentative de la grandeur à estimer. Pour ne pas tomber dans ces travers, il faut introduire des contraintes supplémentaires. Les contraintes que nous utilisons sont de faire tendre la représentation $\mathbf{z}$ vers une moyenne nulle et une variance unitaire. Ces contraintes se traduisent dans les équations par les termes p_t, qui représente une estimation de la moyenne des réponses du premier réseau, et V_t, qui est un coefficient de correction de la variance.

Les modèles d'adaptation des poids sont définis par les équations suivantes, en fonction de l'erreur ϵ_z entre les deux estimation de $\mathbf{z}$:

$$\epsilon_z = \hat{\mathbf{z_2}} - \hat{\mathbf{z_1}} \ , \tag{2.23}$$

$$w^*_{s1} = V_t \left(\hat{\mathbf{z_1}} + \epsilon_z - p_t\right) \ , \tag{2.24}$$

$$w^*_{s2} = \hat{\mathbf{z_2}} - \epsilon_t \ , \tag{2.25}$$

avec :

$$p_{t+1} = p_t + \gamma_p \left(\hat{\mathbf{z_1}} - p_t\right) \ , \tag{2.26}$$

$$V_{t+1} = V_t + \gamma_v V_t \left(1 - (V_t \hat{\mathbf{z_1}})^2\right) \ , \tag{2.27}$$

γ_p et γ_v sont des coefficients d'adaptation, qui peuvent être variables dans le temps. Ces coefficients doivent être assez grands en début d'apprentissage, afin de prendre en compte des variations très grandes des moyennes et variances des poids de la carte de sorties. Au contraire, à la fin de l'apprentissage, ces coefficients doivent être petits, voir nuls, afin de ne pas créer de bruit d'apprentissage, puisque la variance des sorties est quasiment stable.

2.5 Des architectures modulaires en robotique

De nombreuses applications robotiques utilisent la modularité. Nous n'allons pas dresser une liste exhaustive, mais simplement présenter des applications qui mettent en œuvre différentes variantes de modularités mises en avant dans les sections précédentes.

Peu d'applications robotiques utilisent les mélanges d'experts, bien que ceux-ci permettent d'améliorer les performances des réseaux, par exemple, en augmentant la vitesse de convergence du modèle appris (Jacobs et Jordan, 1993). Ils sont aussi utilisés dans des systèmes d'estimation de mouvement pour l'asservissement de robots (Wira, 2002). Chaque expert est en fait un filtre de Kalman adaptatif, le superviseur est quant à lui composé d'un perceptron multicouches.

D'autres architectures proches des mélanges d'experts font l'objet de développements. Gomi et Kawato (1993) utilisent une structure de ce style pour la reconnaissance et la manipulation d'objets. Les deux approches qu'ils proposent se distinguent des mélanges d'experts par le fait que leur superviseur possède un vecteur d'entrées différent de celui des réseaux experts. Cho (2001) utilise également cette forme de modularité. Des contrôleur neuronaux s'occupent du positionnement d'un robot mobile.

Un concept différent de modularité fait l'objet de nombreux travaux au laboratoire. Il s'appuie sur le formalisme du *NeuroModule* (Urban *et al.*, 1997 ; Kuhn, 1997). En fait, un NeuroModule est une structure neuronale qui résulte de l'association hétérogène de deux algorithmes neuronaux. Chacun des deux réseaux possède son propre flux d'entrées. Une couche de Kohonen est associée à une *colonne neuronale*[1] qui peut supporter des fonctionnalités diverses. Une couche de Kohonen sélectionne le neurone vainqueur qui, lui, sélectionne la seule colonne neuronale active qui détermine la sortie. Le NeuroModule est un concept général dans lequel nous retrouvons, par exemple, le SOM–L dont la colonne neuronale est un modèle linéaire local. Les travaux de Buessler et Urban (1998) ; Buessler *et al.* (2002) montrent l'emploi de NeuroModule dans des applications de positionnement, d'asservissement visuel.

La notion d'architecture bidirectionnelle est introduite en 1993 par Kawato et son équipe. Ils associent des modèles directs et des modèles inverses (Wada et Kawato, 1993 ; Miyamoto *et al.*, 1996). Cette association de multiples paires est reprise dans Wolpert et Kawato (1998). Les modèles directs calculent une commande, dont la différence avec la prédiction des modèles inverses sert de signal d'erreur, pour déterminer la part de participation des différentes paires dans le calcul d'une sortie globale. Cette structure se rapproche des mélanges d'experts dans le sens où plusieurs paires apprennent à résoudre une tâche (le rôle des experts) et où la sortie globale est déterminée par un superviseur. Ici, la supervision résulte du calcul d'erreur de prédiction.

L'équipe de Kawato travaille sur la planification de trajectoires (Kawato *et al.*, 1987, 1988 ; Kawato, 1989 ; Kawato *et al.*, 1990) dans le cadre de la commande dynamique. L'architecture MOSAIC (*MOdular Selection and Identification for Control*) (Haruno *et al.*, 2001) a été utilisée en simulation pour le contrôle d'un bras dans quatre

1. Le terme colonne neuronale vient d'une analogie avec le cerveau humain.

contextes exclusifs (le bras manipule un objet parmi quatre possibles, chacun ayant des propriétés mécaniques différentes) (Miall, 2002).

Samejima *et al.* (2003) proposent un apprentissage par renforcement. Cet apprentissage est nécessaire pour la plupart des réseaux en interaction avec l'environnement. Un système de récompense est mis en place, il évalue la manière dont s'est comporté le réseau face à son environnement. C'est au réseau de découvrir quelles sont les réponses qui lui rapportent un maximum de récompenses. Samejima *et al.* (2003) ont appliqué leur apprentissage par renforcement à des tâches de poursuite avec des états cachés et à une tâche de contrôle non-linéaire en temps continu. On retrouvera d'autres applications d'apprentissages par renforcement dans Doya *et al.* (2002).

2.6 Discussions

Ce sont les besoins de l'application qui définissent quelles sont les architectures neuronales à utiliser. La complexité de la tâche à accomplir peut conduire à l'emploi de structures modulaires. Parmi la palette d'outils proposés, il faut faire des choix. Dans cette section, nous allons exposer nos sélections, celles qui vont nous permettre de résoudre nos tâches d'asservissements visuelles.

La fonction qui permet de transformer des informations visuelles en commandes angulaires pour un bras robotique est relativement complexe à définir. Les réseaux de neurones apparaissent comme une solution élégante pour la définition d'un tel modèle. Les réseaux retenus pour accomplir cette tâche doivent posséder un certain nombre de caractéristiques telles que la possibilité d'apprendre des fonctions non-linéaires et compatible avec un apprentissage en ligne.

Les réseaux que nous avons donc choisi d'utiliser sont des SOM–LLM, une variété de cartes de Kohonen étendues dont la sortie est linéarisée avec des modèles linéaires locaux : des ADALINES. Les cartes de Kohonen ont des règles d'apprentissages relativement simples à mettre en œuvre et peu gourmandes en temps de traitement. De plus, ces réseaux utilisent le concept de voisinage, qui leur permet d'apprendre rapidement (principe de généralisation). Ces avantages font de ces réseaux de très bons approximateurs de fonctions qui peuvent être utilisés en ligne.

Cependant, ces réseaux souffrent du problème de la dimensionnalité de l'espace à discrétiser. Des vecteurs d'entrées importants font vite augmenter le coût algorithmique ainsi que le nombre d'exemples d'apprentissages. Un travail d'analyse du problème à traiter permet d'en dégager une décomposition en modules, et donc d'utiliser un ensemble de petits réseaux dédiés, plus faciles à entraîner, plutôt qu'un réseau unique.

Une architecture modulaire permet de réduire localement la dimensionnalité du problème. L'apprentissage de chacun des modules de l'architecture est supervisé. Ils

apprennent chacun une relation non-linéaire liant deux espaces. L'utilisation d'une telle architecture conduit inévitablement à l'introduction de variables intermédiaires. Il faut alors mettre en place des structures d'apprentissages capables de fournir des exemples d'apprentissages à chacun des modules de l'architecture.

L'introduction de modèles inverses dans cette architecture séquentielle va permettre de créer les représentations internes nécessaires à l'apprentissage de chacun des modules. Le fait de devoir estimer des variables intermédiaires rend l'apprentissage plus délicat. Dans cette optique, nous chercherons à ne pas décomposer à outrance. Les modules doivent être les plus simples et les plus pertinents possible. Par exemple, un chaînage du type M2(M1($\mathbf{x}$)) ne contribue pas à créer des modules neuronaux plus simples. Dans un cas de ce genre, toutes les sorties du module M1 servent d'entrées au module M2. M1 traite l'ensemble du vecteur d'entrées, il peut donc réaliser à lui seul les deux étapes de la transformation. Une décomposition n'a de sens que si elle simplifie le problème initial, c'est-à-dire, si elle utilise des modules neuronaux de dimensionnalité plus petite que le problème initial.

2.7 Conclusion

Deux thèmes ont été abordés dans ce chapitre. Le premier concerne les réseaux de neurones, le second concerne les différents aspects de la modularité.

Les réseaux de neurones sont capables d'apprendre à résoudre des tâches dans de nombreux domaines, tels que la classification ou le contrôle par exemple. Une telle diversité de champs applicatifs conduit à des réseaux neuronaux ayant des spécificités propres. Nous trouverons dans ce chapitre une description des différentes architectures neuronales ainsi que des paradigmes d'apprentissage.

Le domaine d'applications, qui nous concerne plus particulièrement, est la robotique. En ce sens, un rapide survol de l'utilisation du connexionnisme à été fait.

Une large place est consacrée aux cartes auto-organisatrices. Ce type particulier de réseaux de neurones a été choisi pour les nombreux attraits qu'il présente.

Il ne faut pas voir les réseaux de neurones comme des solutions miracles à tous les problèmes. Ils constituent en fait de bons outils d'estimation, d'approximation, capables de s'adapter à toutes sortes de situations. Face à des problèmes complexes, un seul réseau n'est souvent pas suffisant, il faut leur préférer une assemblée de réseaux. Nous avons présenté sous le terme de modularité, différentes manières de faire coopérer ou interagir plusieurs réseaux neuronaux. Des exemples d'architectures modulaires en robotique ont servi d'illustrations.

Pour terminer, nous avons expliqué le choix d'utiliser les SOM–LLM dans une structure bidirectionnelle.

Dans les chapitres suivants, nous allons présenter deux simulations mettant en œuvre des réseaux neuronaux dans des applications d'asservissements visuels pour le contrôle de bras robotiques.

3

Application au robot trois axes

Sommaire

Ce chapitre veut mettre en avant les bénéfices que nous avons à utiliser des approches modulaires, pour apprendre des transformations complexes qui lient deux espaces. La décomposition de la tâche à accomplir est le fruit d'une analyse qui permet d'identifier les différentes sous-tâches. Dans l'exemple que nous allons traiter ici, deux modules vont être utilisés.

Nous allons déterminer les valeurs angulaires de chaque articulation d'un robot trois axes, en utilisant deux capteurs visuels. La commande est alors déterminée par une transformation qui lie l'espace des images à l'espace articulaire du robot (chapitre 1). Dans cette application, l'espace image correspond aux coordonnées de l'effecteur ainsi qu'à l'orientation des deux caméras.

Cette transformation peut être obtenue de deux manières : soit par calcul, on parle alors d'approche modèle, soit par apprentissage. Ce sont les approches par apprentissage qui sont utilisées ici.

Plutôt que d'utiliser un contrôleur composé d'un seul réseau de neurones, nous avons choisi de développer un contrôleur composé de plusieurs cartes auto-organisatrices de type SOM–LLM. Le choix d'utiliser ces réseaux de neurones dans une architecture modulaire bidirectionnelle, est une réponse aux contraintes liées aux conditions d'utilisations (temps réel,. . .) et aux problèmes engendrés par la complexité de la tâche à accomplir (dimensionnalité de l'espace d'entrée,. . .)(chapitre 2).

Les modules que nous allons utiliser vont respectivement estimer une configuration alternative à la position de la tête robotique et apprendre la relation qui lie les angles de la tête aux angles du bras.

Dans un premier temps, nous allons présenter notre environnement de travail, avec quelques modèles simples qui vont nous servir de références. Après avoir décrit la tâche robotique à accomplir, nous présenterons les résultats obtenus avec un réseau unique et avec une architecture modulaire. Une partie sera consacrée à la mise en place d'un flux inverse, qui va permettre de fournir les exemples d'apprentissage au premier module. La décomposition d'une tâche n'étant pas unique, nous présenterons, à titre d'exemple, une alternative à la décomposition choisie.

3.1 L'environnement de simulation

Dans ce chapitre, nous allons montrer comment il est possible d'apprendre la relation complexe qui permet de déterminer la configuration angulaire d'un bras robotique à trois axes en utilisant les informations visuelles. Apprendre une telle relation n'est pas trivial. Cela nécessite l'emploi d'architectures modulaires et de structures bidirectionnelle. Ce travail a été initié par Jean-Luc Buessler (1999) et s'étend à l'utilisation de cartes auto-organisatrices de type SOM–LLM pour des systèmes robotiques dans l'espace tridimensionnel.

Avant de développer plus en avant la tâche robotique à effectuer, nous allons présenter l'environnement de travail.

La scène robotique que nous allons utiliser dans nos simulations est une réplique de la plate-forme expérimentale que nous possédons au laboratoire.

Dans cette section, nous allons présenter les différents éléments qui composent notre scène robotique, à savoir :

- le robot trois axes ;
- la tête robotique ;
- et la scène (avec notamment la disposition des différents éléments).

Mais avant toutes choses, nous allons faire un bref rappel sur les coordonnées homogènes.

> Les solutions proposées aux applications qui vont suivre ne sont pas liées à une configuration particulière de la scène robotique. Cependant, pour illustrer notre propos, nous allons utiliser des valeurs numériques propres à notre plate-forme. Elles seront signalées par une mise en forme particulière, qui est celle de ce paragraphe.

α.– Les coordonnées homogènes

Un corps solide est caractérisé par une position et une orientation dans un repère donné. Le déplacement de cet objet dans l'espace revient à changer sa position et son orientation. Les deux opérateurs utilisés sont la rotation qui peut être formalisée par une matrice $\mathbf{A}^{3\times3}$, et la translation qui peut être formalisée par un vecteur $\mathbf{t}$ de taille trois. Ces deux transformations correspondent à un changement de repère et peuvent être réunies dans une matrice $\mathbf{T}^{4\times4}$ qui est la matrice des coordonnées homogènes (Paul, 1981).

La matrice $\mathbf{A}$ qui exprime l'orientation peut être formalisée par les angles d'Euler ψ, θ et ϕ. En effet le théorème de rotation d'Euler dit que toute rotation peut être définie en utilisant trois angles. Si chacune des trois rotations est écrite en terme de

matrice rotation ($\mathbf{\Psi}$, $\mathbf{\Theta}$, $\mathbf{\Phi}$), alors la matrice générale $\mathbf{A}$ peut être :

$$\mathbf{A} = \mathbf{\Psi\Theta\Phi} \ . \tag{3.1}$$

Chacune des trois rotations peut être effectuée autour de n'importe quel axe x, y ou z. Une rotation de ψ radians autour de l'axe x est définie de la manière suivante :

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \ . \tag{3.2}$$

Une rotation de ψ radians autour de l'axe y est définie de la manière suivante :

$$R_y(\psi) = \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix} \ . \tag{3.3}$$

Une rotation de ψ radians autour de l'axe z est définie de la manière suivante :

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \ . \tag{3.4}$$

Considérons les deux repères : $\mathbf{r_r}$, qui pourrait correspondre au référentiel du robot, et $\mathbf{r_c}$, qui pourrait correspondre au référentiel d'une caméra, d'origines respectivement O_r et O_c. Un point P, comme le poignet par exemple, peut être référencé dans les deux repères avec $\mathbf{x_r}$ le vecteur coordonnées dans $\mathbf{r_r}$ et avec $\mathbf{x_c}$ le vecteur coordonnées dans $\mathbf{r_c}$. La relation entre les deux vecteurs $\mathbf{x_r}$ et $\mathbf{x_c}$ s'écrit :

$$\mathbf{x_r} = \mathbf{A}\mathbf{x_c} + \mathbf{t} \ , \tag{3.5}$$

avec $\mathbf{t}$ les coordonnées de O_c dans le repère $\mathbf{r_r}$, et $\mathbf{A}$ la matrice de rotation du repère $\mathbf{r_r}$ par rapport à $\mathbf{r_c}$.

La matrice de transformation homogène est :

$$\mathbf{T} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \ . \tag{3.6}$$

En utilisant $\mathbf{T}$ l'équation (3.5) devient :

$$\mathbf{X_r} = \mathbf{T}\mathbf{X_c} \ , \tag{3.7}$$

où $\mathbf{X_r} = [\mathbf{x_r} \quad 1]^T$ et $\mathbf{X_c} = [\mathbf{x_c} \quad 1]^T$ sont respectivement les coordonnées homogènes du point P dans $\mathbf{r_r}$ et $\mathbf{r_c}$.

Étant donné que la matrice $\mathbf{T}$ est inversible, la transformation qui permet de passer du repère $\mathbf{r_c}$ au repère $\mathbf{r_r}$ est donnée par l'équation suivante :

$$\mathbf{X_c} = \mathbf{T^{-1}X_r} \ . \tag{3.8}$$

Par exemple, pour le passage du repère du robot au repère de la caméra, nous définissons la matrice $\mathbf{A}$ comme la composition d'une rotation ψ suivant l'axe z, d'une rotation θ suivant l'axe x et d'une rotation ϕ suivant l'axe y.

$$\mathbf{A} = R_z(\psi)\, R_x(\theta)\, R_y(\phi) \ . \tag{3.9}$$

Ici :

- ψ est fonction de l'angle panoramique α_p ;
- θ est fonction de l'angle azimutal (le tilt) α_t ;
- ϕ est fonction de l'angle de vergence α_v.

Le vecteur $\mathbf{t}$ correspond aux coordonnées de la caméra exprimées dans le repère de départ, celui du robot. Ces coordonnées dépendent de la position de la base de la tête et des angles panoramique et de vergence, qui déplacent les plans images autour de la base.

β.– Le robot trois axes

Le robot trois axes que nous utilisons dans nos simulations est une version simplifiée de Sylvia, le robot de la plate forme expérimentale (voir figure 3.1), dans le sens ou nous ne conservons que les trois premiers axes. Le premier axe permet la rotation de l'ensemble, les angles θ_2 et θ_3 sont dans le même plan. Ce robot peut atteindre n'importe quelle position dans l'espace 3D.

Le modèle du robot pour les simulations est défini par les équations suivantes :

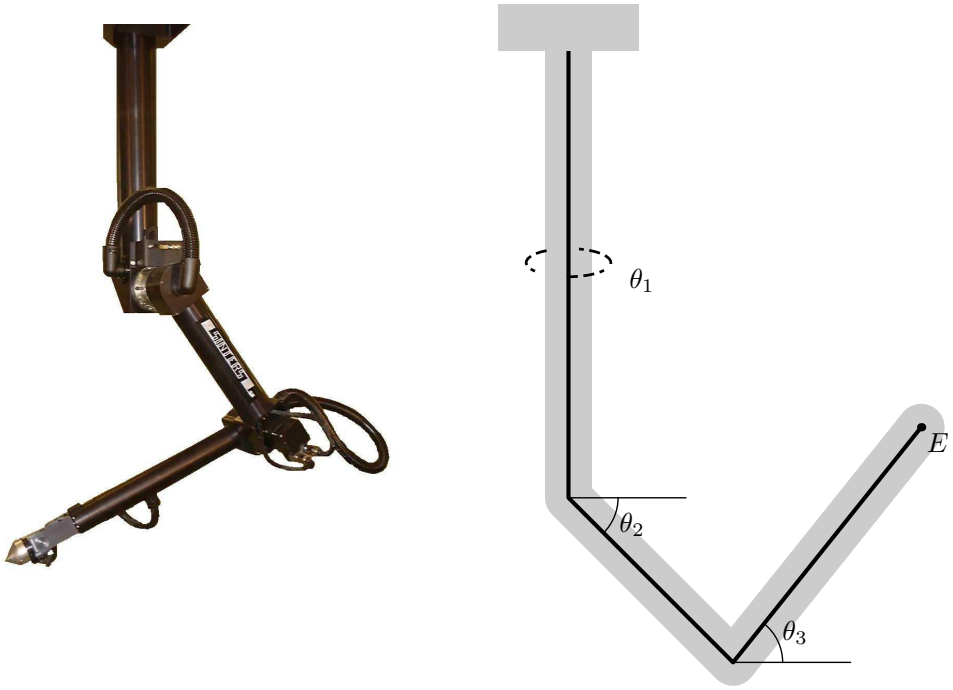
$$x = (\mathrm{l}_2 \cos\theta_2 + \mathrm{l}_3 \cos\theta_3)\cos\theta_1 + \mathrm{x_b} \ , \tag{3.10}$$

$$y = (\mathrm{l}_2 \cos\theta_2 + \mathrm{l}_3 \cos\theta_3)\sin\theta_1 + \mathrm{y_b} \ , \tag{3.11}$$

$$z = \mathrm{l}_2 \sin\theta_2 + \mathrm{l}_3 \sin\theta_3 - \mathrm{l}_1 + \mathrm{z_b} \ . \tag{3.12}$$

l_1, l_2 et l_3 sont respectivement les longueurs des axes un, deux et trois et valent 680 mm, 400 mm et 480 mm.

θ_1 est l'angle de l'axe un, θ_2 et θ_3 sont les angles formés par les axes deux et trois par rapport à l'horizontal. $\mathrm{x_b}$, $\mathrm{y_b}$ et $\mathrm{z_b}$ sont les coordonnées de la base du bras.



a) b)

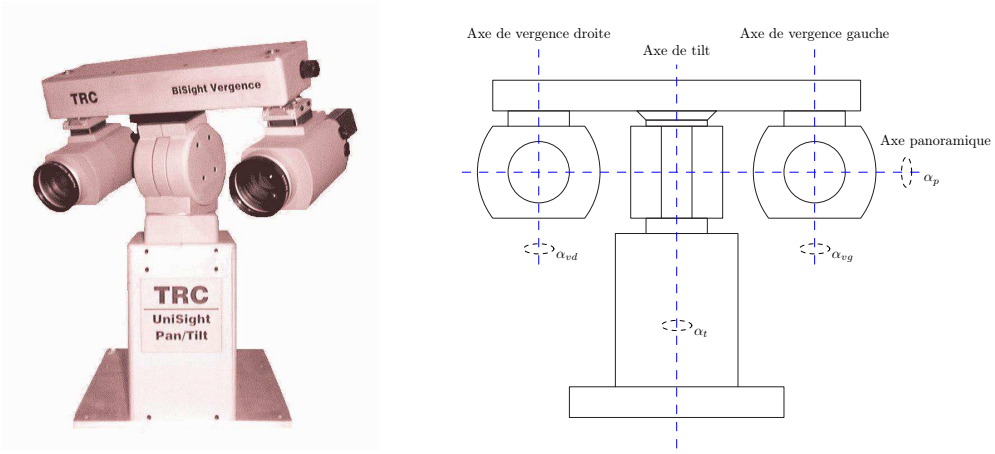
Fig. 3.1 *Le bras robotique*

À gauche, une photo de Sylvia, le robot de la plate-forme expérimentale. À droite, une vue schématique simplifiée de ce même bras. Seuls les trois premiers axes sont utilisés dans cette première simulation. θ_1, θ_2 et θ_3 sont les trois angles à déterminer. Ils sont définis par rapport à l'horizontale. E représente l'effecteur.

γ.– La tête robotique

Les informations relatives au bras peuvent être fournies par des capteurs internes (proprioceptifs). Il est possible de connaître par exemple les positions et vitesses angulaires. Si nous voulons tenir compte de l'environnement dans lequel le bras manipulateur évolue, il faut alors utiliser des capteurs externes (extéroceptifs), comme des caméras, qui eux renseignent sur la présence, la distance avec d'autres objets de la scène comme une cible ou un obstacle par exemple. Ces capteurs sont indispensables dans les tâches d'asservissements visuels.

De nombreuses informations sont disponibles dans les images et toutes ne sont pas forcément utiles. L'extraction des données importantes pour un problème donné est souvent facilité en marquant les objets d'intérêts. Ces marqueurs peuvent être de différentes natures comme des diodes, des cibles, des motifs ... Pour nos premières manipulations, l'effecteur du bras robotique était repéré par une diode. Un simple seuillage sur l'image permettait de repérer facilement l'extrémité du robot. La complexité des nouvelles tâches et des tâches futures nous poussent à utiliser des marqueurs plus complexes en raison du nombre plus important d'objets à identifier



a) b)

Fig. 3.2 *La tête robotique*

À gauche, une photo de la tête robotique de la plate-forme expérimentale. À droite, une vue schématisée de cette même tête. Les différents axes de rotations y sont représentés. Ils permettent d'orienter les deux caméras.

dans les images (poignet, effecteur, cible, ...). Les travaux récents se penchent sur l'utilisation de cibles couleurs. La contrainte principale est due au fait que l'extraction des informations doit être effectuée par un traitement d'image peu coûteux en temps, puisqu'il doit être effectué en temps réel. Ici, le temps réel correspond à la cadence vidéo.

La tête robotique

Les caméras que nous utilisons sont placées sur un système mécanique robotisé (voir la figure 3.2). Chacune des deux caméras peut être orientée de manière à ce que toutes les régions d'intérêts soient simultanément dans le champs de vision. L'orientation des caméras se fait à l'aide de quatre axes : l'axe panoramique qui permet de balayer la scène dans le sens horizontal, l'axe azimutal (de tilt) qui permet de balayer la scène dans le sens vertical et deux axes de vergences propres à chacune des deux caméras. Les axes de rotation des axes de vergences sont parallèles à l'axe panoramique.

Les positions des centres des plans images de chacune des caméras ne dépendent que de la position de la tête robotique dans l'espace 3D et de l'angle panoramique. En revanche, les angles de tilt et de vergences permettent d'orienter chacun de ces plans images. Les équations suivantes donnent les positions des centres des plans images.

$$\begin{bmatrix} x_{cg} & y_{cg} & z_{cg} \end{bmatrix} = \begin{bmatrix} \mathrm{x_t} - \mathrm{B}\sin\alpha_\mathrm{t} & \mathrm{y_t} - \mathrm{B}\cos\alpha_\mathrm{t} & \mathrm{z_t} \end{bmatrix} , \tag{3.13}$$

$$\begin{bmatrix} x_{cd} & y_{cd} & z_{cd} \end{bmatrix} = \begin{bmatrix} \mathrm{x_t} + \mathrm{B}\sin\alpha_\mathrm{t} & \mathrm{y_t} + \mathrm{B}\cos\alpha_\mathrm{t} & \mathrm{z_t} \end{bmatrix} . \tag{3.14}$$

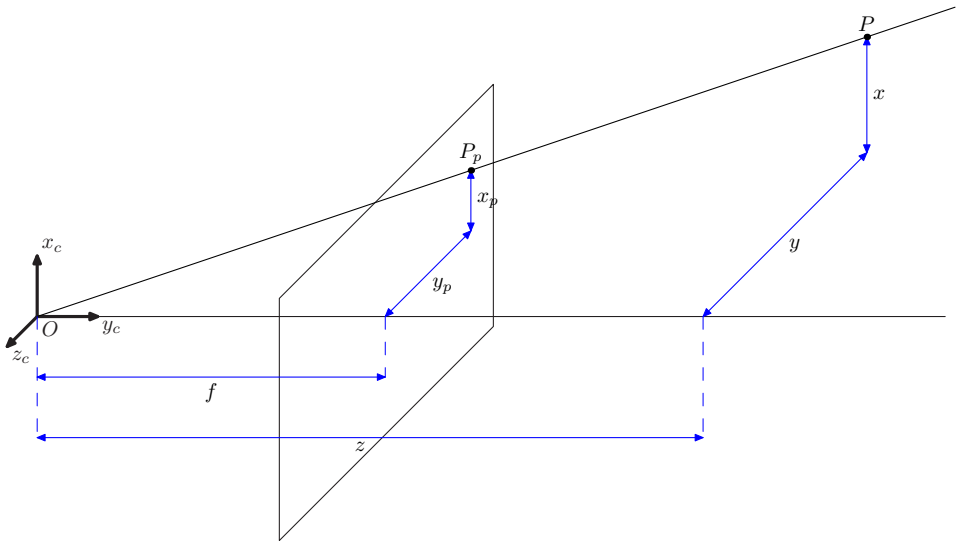


Fig. 3.3 *Projection perspective*

Dans ces équations x_{cg}, y_{cg} et z_{cg} sont les coordonnées du centre du plan image de l'image gauche, x_{cd}, y_{cd} et z_{cd} sont les coordonnées du centre du plan image de l'image droite, x_t, y_t et z_t les coordonnées de la base de la tête, B la demi-distance entre les deux plans et α_t l'angle de tilt.

Modèle de caméra

Le modèle de caméra que nous avons retenu est celui dit du « trou d'épingle » (*pinhole* en anglais). Il s'agit en fait de faire la projection perspective d'un élément observé de l'espace 3D sur le plan image (voir la figure 3.3).

Le point P observé, de coordonnées $\{x,y,z\}$ dans le repère (O,x_c,x_c,z_c) de la caméra, se projette en P_p de coordonnées $\{x_p,y_p\}$ dans le plan image. Le plan image se situe à la distance f, la distance focale. En utilisant le théorème de Thalès, nous obtenons :

$$x_p = \frac{fx}{z} \quad , \tag{3.15}$$

$$y_p = \frac{fy}{z} \quad . \tag{3.16}$$

Les valeurs x_p et y_p sont exprimées en millimètres dans le plan image. Dans le cas des caméras, ce plan image est un capteur CCD (*Charge Coupled Device*). Il faut alors exprimer x_p et y_p en pixels. Les matrices CCD utilisées sont composées de 640 colonnes de 11 μm de largeur et de 480 lignes de 13 μm de hauteur.

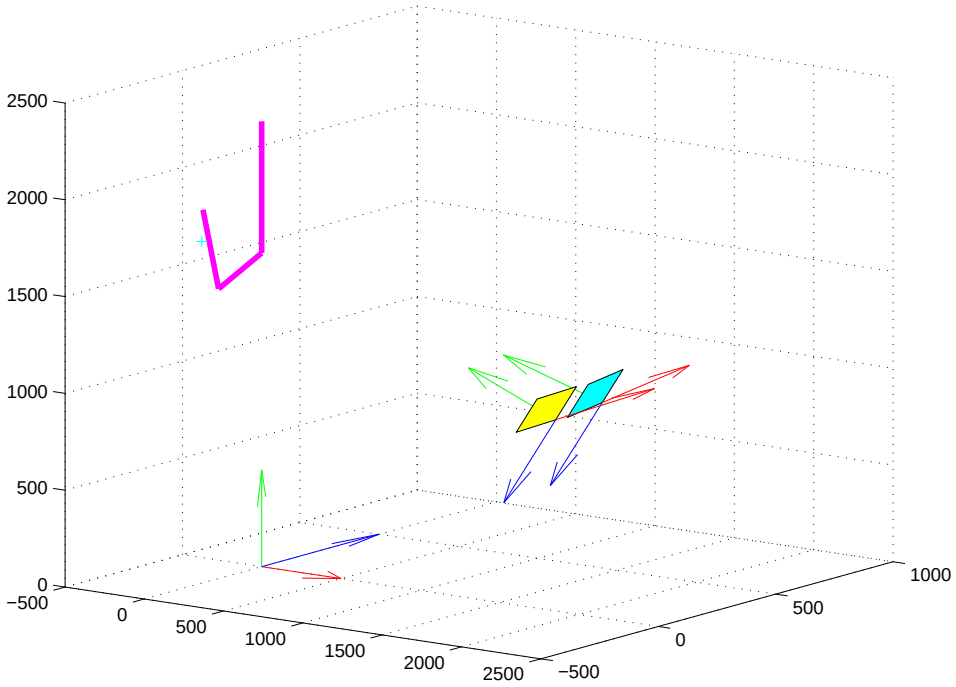


Fig. 3.4 *Environnement de simulation*

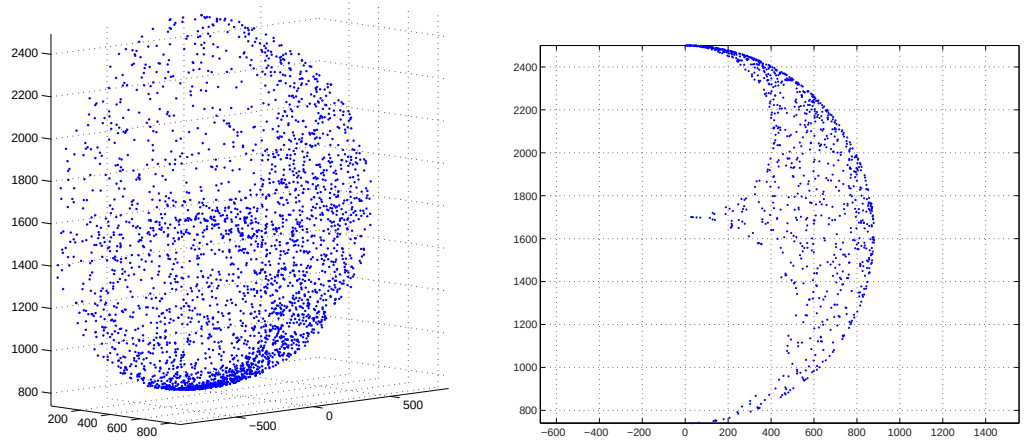
En magenta le robot trois axes suspendu au plafond, en jaune la caméra gauche, en bleu la caméra droite associés à leurs repères respectifs. Les repères sont définis comme suit : en rouge l'axe x, en bleu l'axe y et en vert l'axe z.

δ.– La scène

La scène de simulation que nous allons utiliser est composée d'un bras robotique et d'une tête surmontée de deux caméras. Ces deux « acteurs » seront disposés dans un environnement proche de la plate-forme réelle. La figure 3.4 donne un aperçu de l'environnement de simulation. Nous pouvons y voir le bras (en magenta) et les deux plans images (en jaune pour le gauche et en bleu pour le droit). Chacun de ces trois éléments est associé à son propre repère. Les axes des repères sont mis en évidence par leurs couleurs : rouge pour l'axe x, bleu pour l'axe y et vert pour l'axe z. Le passage d'un repère à l'autre est définit par la matrice des coordonnées homogène que nous avons évoqué au début de cette section.

> La base du bras robotique a les coordonnées $\begin{bmatrix} 0 & 0 & 2300 \end{bmatrix}$ (le robot est fixé au plafond). Dans ce même repère, celui lié au bras, la base de la tête robotique a les coordonnées $\begin{bmatrix} 2000 & 0 & 1050 \end{bmatrix}$. Ces coordonnées sont exprimées en millimètres.

Le robot tel que nous l'avons défini peut atteindre quasiment toutes les positions 3D situées dans une sphère qui a pour centre le coude du bras (liaison entre les deux premiers élements) et pour rayon la somme des longueurs des axes deux et trois. En réalité, cette enveloppe est tronquée par le plafond et les limites angulaires dues à



a) b)

Fig. 3.5 *Distribution de l'effecteur dans l'espace 3D*

la mécanique. Ceci n'est pas gênant dans le cas de notre apprentissage. Nous avons cependant ajouté d'autres contraintes sur les angles θ de chacune des articulations, de telle sorte à ce que l'effecteur se limite à des x positifs (le bras se déplacera donc dans une demi-sphère). De plus, les angles seront choisis de telle sorte que chacune des positions de l'espace 3D ne soit atteignable que d'une seule et unique façon.

Les contraintes angulaires du bras peuvent se résumer de la manière suivante :

- $-\frac{\pi}{2} \leq \theta_1 \leq \frac{\pi}{2}$;
- $-\frac{\pi}{2} \leq \theta_2 \leq \frac{\pi}{2}$;
- $-\frac{\pi}{2} \leq \theta_3 \leq \frac{\pi}{2}$ avec $\theta_2 \geq \theta_3$.

La figure 3.5 a) montre un certain nombre de positions atteignables par l'effecteur du robot. La figure 3.5 b) montre les positions de l'effecteur pour un angle θ_1 nul. Ce sont ces positions qui sont utilisées pour les apprentissages. Il faut remarquer que la distribution n'est pas uniforme dans l'espace 3D, car en effet, seuls les angles θ ont une distribution uniforme. Notons que cette configuration est valable pour l'apprentissage mais n'est sans doute pas très réaliste. En effet, nous trouverons plutôt dans la pratique des robots qui travaillent dans une zone plus restreinte, éloignée des positions extrêmes atteignables. Nous avons délibérément choisi une zone plus complexe afin de tester les capacités de notre apprentissage.

Des contraintes peuvent aussi être appliquées aux angles α de la tête. Il reste néanmoins nécessaire que l'effecteur se projette sur les plans images de chacune des deux caméras.

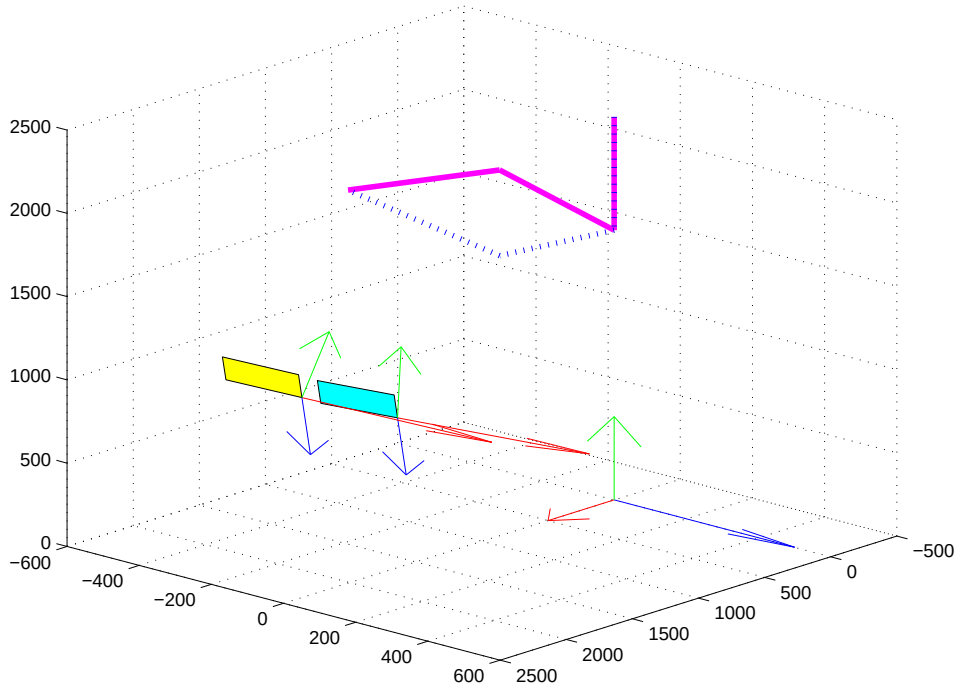


Fig. 3.6 *Exemple de redondance dans l'espace de sortie*

Dans cet exemple, le robot trois axes atteint une position 3D de deux manières différentes (la première en trait plein, la seconde en pointillés). Pour que l'apprentissage soit efficace, les redondances ne doivent pas être négligées.

3.2 Le traitement des redondances

La qualité d'un apprentissage neuronal est influencée par un certain nombre de critères, comme la dimentionnalité de l'espace d'entrées, le choix des paramètres d'apprentissage ou encore par le choix des exemples d'apprentissage.

Les réseaux de neurones que nous utilisons ont un apprentissage supervisé. C'est-à-dire qu'il faut leur fournir, pendant la phase d'apprentissage, une entrée et une sortie désirée. Les exemples d'apprentissage que nous utilisons sont choisis de manière à éviter les redondances.

La redondance correspond à l'association de plusieurs sorties différentes à une seule et unique entrée. Par exemple, sur la figure 3.6, la position qu'occupe l'effecteur du robot peut être obtenue de deux manières différentes. La première est symbolisée par le trait plein et la seconde par des pointillés. Deux configurations articulaires correspondent à une seule position 3D.

Nous mettons en place suffisamment de contraintes pour qu'il n'y ait plus de cas de redondances. Celles-ci peuvent perturber fortement l'apprentissage. Si nous nous intéressons au cas d'une carte de Kohonen étendue, la modification des poids de

sorties est régie par les équations (2.8) et (2.9), que nous rappelons ici :

$$w_r^{out} = w_r^{out} + \Delta w_r^{out} \ ,$$

avec :

$$\Delta w_r^{out} = \mu' \cdot h'_{rs} \cdot (\mathbf{y}_k - w_r^{out}) \ ,$$

avec $\mathbf{y}_k$ la sortie désirée. Nous pouvons comparer w_r^{out} à une moyenne pondérée des différentes sorties désirées, $\mathbf{y}_k$, présentées. Si nous reprenons l'exemple de la figure 3.6, cela signifie que si les deux cas, en trait plein et en pointillés, sont présentés avec la même probabilité, alors les angles θ_2 et θ_3 vont naturellement tendre vers zéro (moyenne des deux cas) plutôt que vers les valeurs réelles.

La contrainte que nous appliquons à notre bras robotique est que l'angle θ_2 doit toujours être supérieur à l'angle θ_3. Un traitement différent aurait pu être choisi, comme par exemple imposer une orientation (atteindre la cible par le haut, etc.).

3.3 Détermination de la tâche à accomplir

Nous voulons montrer que l'emploi de réseaux de neurones au sein d'une architecture modulaire permet d'apprendre des fonctions complexes. Nous entendons par fonctions complexes, des fonctions qui dépendent d'un grand nombre d'entrées.

Les architectures modulaires ont été présentées dans le chapitre 2 et nous allons, ici, en mettre une en œuvre. Afin de bien comprendre les avantages à utiliser une telle architecture, nous allons comparer les résultats obtenus à une approche plus classique, qui est l'emploi d'un réseau unique.

Nous allons nous appuyer sur l'application suivante pour illustrer notre propos. Il s'agit d'apprendre le modèle qui permet de déterminer la configuration angulaire d'un robot trois axes en utilisant les informations visuelles (configuration articulaire de la tête et coordonnées de l'effecteur dans les images).

Pour ce faire, l'effecteur du bras manipulateur est observé par la tête robotique sur laquelle sont montées deux caméras. Dans cette application, seuls les angles de tilt et de vergences seront utilisés. L'axe panoramique, α_p restera nul.

Les entrées du système, fournies par les différents capteurs sont :

- α_t l'angle de tilt, α_{vg} et α_{vd} les angles de vergences respectivement gauche et droite ;
- I_{xg} et I_{yg} les coordonnées de l'effecteur dans l'image gauche, respectivement suivant l'axe x et suivant l'axe y ;

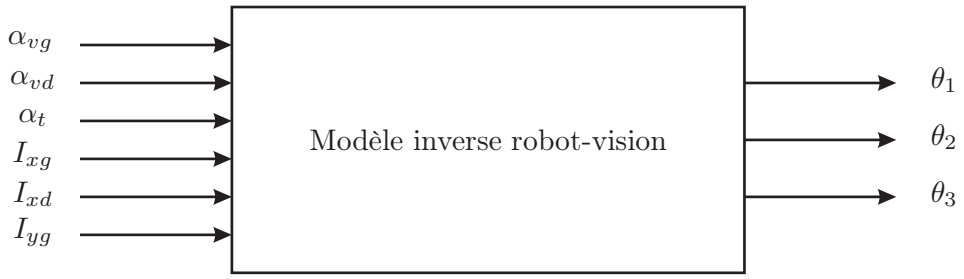


Fig. 3.7 *Tâche à accomplir*

Tâche à accomplir : il s'agit de déterminer la configuration angulaire d'un robot trois axes en utilisant les informations visuelles.

– I_{xd} et I_{yd} les coordonnées de l'effecteur dans l'image droite, respectivement suivant l'axe x et suivant l'axe y.

Les grandeurs à estimer sont les trois angles du bras robotique θ_1, θ_2 et θ_3.

La transformation à apprendre est symbolisée sur la figure 3.7, il s'agit en fait d'apprendre le modèle inverse d'un système robot–vision.

Les réseaux de neurones que nous utilisons sont des SOM-LLM. Ils ont été choisis pour leur capacité à apprendre en ligne.

3.4 L'APPRENTISSAGE « CLASSIQUE »

Dans cette section, nous allons apprendre le modèle inverse robot-vision, que nous avons décrit plus haut. Le contrôleur est composé d'un réseau de neurones unique de type SOM–LLM. Dans le cadre de l'apprentissage, cette solution parait être la plus simple.

α.– Choix des entrées / sorties

Dans le cas de l'utilisation d'un réseau de neurones unique, définir les entrées ne pose guère de problème. En effet, le vecteur d'entrées est composé de toutes les grandeurs qui participent au calcul de la sortie. Bien sûr, une mise en forme des entrées peut et doit être opérée. Chacune des entrées doit être normalisée. En effet, la sélection du neurone vainqueur dans la grille d'entrées d'une carte auto-organisatrice se fait en calculant une distance qui dépend de toutes les grandeurs du vecteur d'entrées. Si les grandeurs ne sont pas comparables en termes de valeur, de propriété statistique, certaines d'entre elles pourraient alors ne pas participer à l'apprentissage et devenir inutiles. Pour palier à une telle situation, il faut normaliser les grandeurs du vecteur d'entrées pour que chacune d'entre elles participe à l'élection du neurone

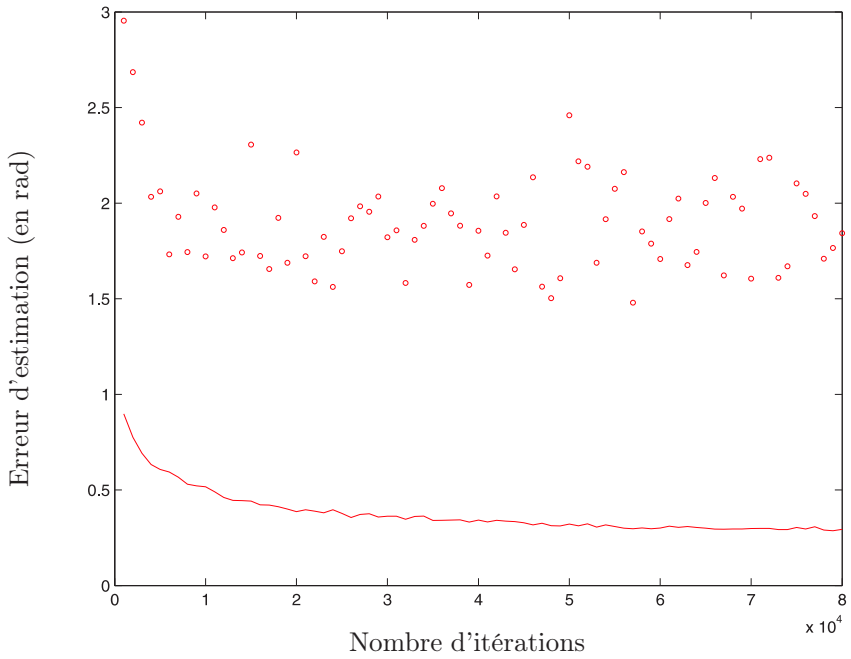


Fig. 3.8 *Résultat d'un apprentissage à carte unique*

L'erreur moyenne est représentée pas la ligne continue, les erreurs maximum sont représentées avec des « o ». Les erreurs sont exprimées en radians.

vainqueur. Ainsi l'apprentissage ne sera pas faussé. Cette normalisation peut être faite de diverses manières. Nous avons choisi de faire tendre la moyenne de chaque élément vers zéro, avec une variance unitaire.

Dans cette application robotique, le modèle à apprendre est fonction de six entrées : les trois angles α_{vg}, α_{vd}, α_t de la tête ainsi que les abscisses de l'effecteur dans les deux images et une ordonnée, I_{yg} ou I_{yd} (les ordonnées de l'effecteur dans les deux images sont quasiment identiques).

Les sorties désirées sont les trois angles du bras robotique.

β.– Résultats de l'apprentissage et conclusion

L'apprentissage du modèle inverse robot-vision, avec un réseau de neurones unique ne donne pas satisfaction. L'erreur d'apprentissage est montrée sur la figure 3.8. Bien que l'erreur moyenne diminue au cours du temps, elle reste élevée. Nous constatons par ailleurs, que les erreurs maximum n'évoluent guère.

Les résultats obtenus pour la modélisation inverse d'un système robot–vision avec un réseau unique sont assez médiocres. Cela confirme un certain nombre d'a priori.

La dimensionnalité du vecteur d'entrées est un facteur déterminant quant à l'efficacité et la faisabilité d'un apprentissage neuronal à base de cartes auto-organisatrices. Des cartes de dimension une, deux ou trois ne posent pas de problème particulier, et

leur convergence est assurée sous réserve que le modèle à apprendre existe. Plus un vecteur d'entrées comporte de grandeurs, plus la paramétrisation et l'apprentissage sont difficiles, comme nous le montrons dans l'exemple traité.

Un autre problème induit par la complexité de la tâche à apprendre est la taille des réseaux utilisés. Plus la taille du réseau est grande, plus le déploiement des cartes est difficile (multiplication des accidents) et plus les temps de calculs augmentent, etc.

La solution présentée pour palier à ces problèmes est l'emploi de plusieurs cartes de dimensions réduites. La tâche complexe va être décomposée en une succession de tâches plus petites, plus faciles à mettre en œuvre.

3.5 L'APPRENTISSAGE MODULAIRE

Nous avons déjà évoqué dans le chapitre 2 que la décomposition en modules est une solution alternative élégante qui permet de réduire la dimensionnalité de l'espace d'entrées.

Plutôt que d'utiliser un réseau complexe, que nous ne savons pas forcément entraîner, nous allons plutôt utiliser plusieurs réseaux de taille plus modeste, dont l'apprentissage est relativement bien maîtrisé. Un problème complexe va être scindé en plusieurs sous-problèmes avec pour but de réduire la dimensionnalité de chaque réseau de neurones. Les conséquences directes d'une dimensionnalité moindre sont une réduction exponentielle du temps de traitement et de l'espace mémoire utilisé par les processeurs.

L'utilisation de plusieurs cartes de petite taille permet de prendre en compte un grand nombre de variables. L'approche n'est plus générale, elle se base sur un travail d'analyse de la fonction à apprendre. Cette démarche est certes plus délicate à mettre en œuvre, mais elle s'avère être très puissante.

α.– La décomposition

Tout l'enjeu de la décomposition va consister en l'élaboration de petits modules, que nous limiterons à trois entrées maximum, capables de mimer le modèle inverse robot–vision.

L'effecteur est observé par deux caméras dont les orientations sont quelconques. La seule contrainte est que l'effecteur du robot se projette dans les images droite et gauche. L'effecteur se projette donc à une position quelconque dans chacune des images. Notre décomposition repose sur une configuration alternative des orientations des caméras. Elle consiste à déterminer les nouveaux angles de la tête de manière à ce que l'effecteur se projette aux centres des deux images.

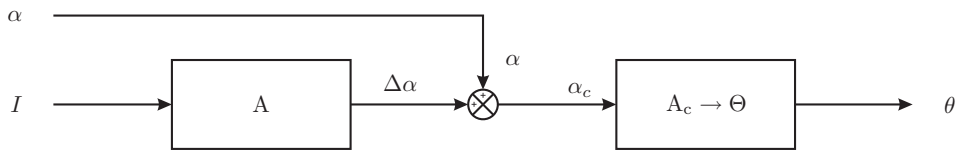


Fig. 3.9 *Architecture modulaire pour le modèle inverse robot–vision*

Avec α les angles de la tête, I les informations images, $\Delta\alpha$ les modifications angulaires pour centrer l'effecteur dans les images, α_c les angles de la tête qui centrent l'effecteur dans les images et θ les angles du bras robotique.

Pour une position de P donnée, il n'existe qu'une seule manière de positionner la tête pour que la projection de P soit centrée dans chacune des images. La relation qui lie un triplet de θ à un triplet d'α est alors bijective. Elle peut par conséquent être apprise par un module.

La décomposition que nous retenons pour apprendre le passage des informations de la tête (c'est-à-dire des angles de la tête et des coordonnées dans les images) aux angles du bras robotique est composée de deux modules :

- Un premier module est dédié au calcul de la configuration alternative de la tête qui permet de centrer la projection de l'effecteur dans les deux images. Ce premier module lui-même peut être scindé en trois modules, un pour chacun des angles α.
- Un second module va apprendre la relation qu'il y a entre les angles de la tête centrée sur un point et la position angulaire du bras robotique. Ce modèle inverse robot–vision est fortement simplifié par rapport au même modèle présenté dans la section précédente (c'est un module qui n'a plus que trois entrées).

Le schéma de décomposition utilisé est présenté sur la figure 3.9. Nous allons maintenant présenter l'apprentissage de chacun des modules.

β.– **Apprentissage de** A

Le module A apprend la relation qu'il y a entre l'espace image et la commande angulaire à appliquer à la tête pour que l'objet regardé soit projeté aux centres des images.

Sur la figure 3.10, nous pouvons voir en noir les deux axes focaux pour une position angulaire de la tête donnée. P est l'effecteur du robot observé. En bleu, nous avons la position désirée des deux axes focaux. Les deux angles $\Delta\alpha_{vg}$ et $\Delta\alpha_{vd}$ sont les deux commandes angulaires à appliquer pour centrer l'effecteur selon l'axe x dans les deux images. De la même manière, il existe une commande $\Delta\alpha_t$ pour centrer l'effecteur selon l'axe y.

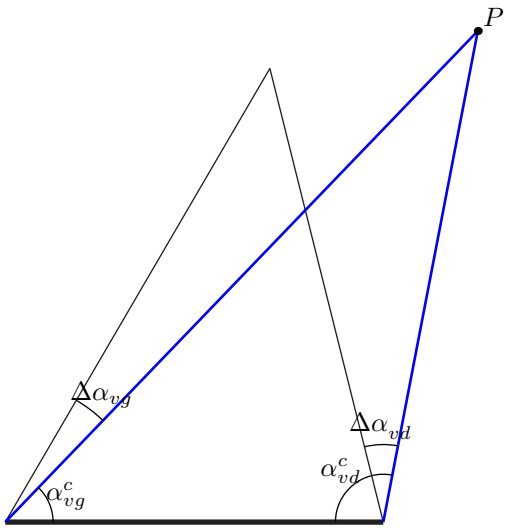


Fig. 3.10 *Modification des axes focaux pour centrer l'effecteur dans les images*

La commande la plus facile à déterminer est $\Delta\alpha_t$, elle ne dépend que d'une information visuelle I_x. L'équation suivante donne la relation à apprendre entre $\Delta\alpha_t$ et I_x :

$$\Delta\alpha_t = \tan^{-1}\left(\frac{I_{yg}\mathrm{dy}}{f}\right) \quad , \tag{3.17}$$

avec dy la taille d'un pixel suivant l'axe y (de la même manière dx est la taille du pixel suivant x).

Pour être tout à fait rigoureux, il faudrait faire la différence entre un $\Delta\alpha_{tg}$ pour la caméra gauche et un $\Delta\alpha_{td}$ pour la caméra droite. La conception de la tête ne le permet pas. Cette différence est induite par la différence existant entre I_{yg} et I_{yd} qui, elle, dépend des distances qui séparent l'effecteur des deux caméras. Nous allons, dans un premier temps, considérer que cette différence est négligeable puisque la distance entre les deux caméras est petite devant les distances caméras–effecteur. Pour que $I_{yg} = I_{yd}$, il faut que les deux angles de vergences soient égaux (cas de vergences symétriques).

Les deux commandes $\Delta\alpha_{vg}$ et $\Delta\alpha_{vd}$ dépendent elles aussi d'informations visuelles, respectivement I_{xg} et I_{xd}.

$$\Delta\alpha_{vg} = \tan^{-1}\left(\frac{I_{xg}\mathrm{dx}}{f}\right) \quad , \tag{3.18}$$

$$\Delta\alpha_{vd} = \tan^{-1}\left(\frac{I_{xd}\mathrm{dx}}{f}\right) \quad . \tag{3.19}$$

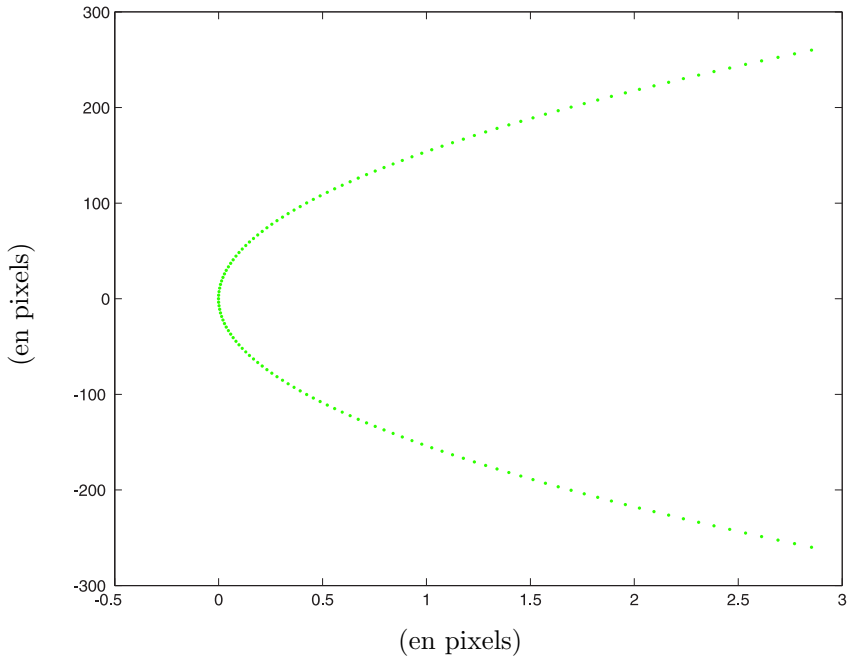


Fig. 3.11 *Position de l'effecteur dans une image pour différents α_t*

Nous montrons sur cette figure que l'axe de vergence est lié à l'axe de tilt. Nous constatons que pour une vergence fixe et différentes valeurs d'angle de tilt, l'effecteur se projette le long d'une parabole, dans les images. Les grandeurs sont exprimées en pixels.

Il faut cependant bien garder à l'esprit que les plans de rotation suivant les axes de vergences dépendent de la position de l'axe de tilt. Pour preuve, la figure 3.11 montre les positions d'un effecteur dans une image pour un angle de vergence fixe et plusieurs valeurs pour l'angle de tilt.

Les fonctions à apprendre du module A restent sommes toutes relativement simples. L'emploi de trois réseaux 1D est largement suffisant. Les réseaux destinés à l'apprentissage des $\Delta\alpha_v$ auront deux entrées, l'information image I_x et l'angle α_t correspondant, alors que celui destiné à l'apprentissage de $\Delta\alpha_v$ n'en n'aura qu'une, I_y. Les tables 3.1 et 3.2 donnent une idée des paramètres d'apprentissage des réseaux de neurones à entraîner.

Quelques 5000 exemples d'apprentissage sont suffisants pour déployer les cartes d'entrées et de sorties ainsi que pour apprendre les jacobiennes associées. À la suite de cet apprentissage, une série de 1000 nouveaux exemples sont présentés aux trois modules. Les sorties calculées – ou estimées – sont comparées aux sorties réelles. L'erreur moyenne d'estimation en pourcentage est très proche de zéro (de l'ordre de 10^{-2}) et l'erreur maximale d'estimation en pourcentage avoisine le pour cent.

Ces résultats ne sont pas exceptionnels. Ils s'expliquent par le fait que la fonction à mimer est quasi-linéaire. Les sorties de ce module A, très satisfaisantes, vont être associées aux valeurs courantes des angles de la tête pour former les angles qui

centrent l'effecteur dans les deux images.

$$\alpha^c_{vg} = \alpha_{vg} + \Delta\alpha_{vg} \ , \tag{3.20}$$

$$\alpha^c_{vd} = \alpha_{vd} + \Delta\alpha_{vd} \ , \tag{3.21}$$

$$\alpha^c_t = \alpha_t + \Delta\alpha_t \ . \tag{3.22}$$

Les nouveaux angles de la tête, les α^c, sont aussi représentatifs que les angles α associés aux informations images. Nous sommes passés d'un espace $\in \mathbb{R}^6$ à un espace $\in \mathbb{R}^3$. Les angles ainsi calculés vont servir d'entrées au module suivant, le module $\mathrm{A_c} \rightarrow \Theta$.

Cette configuration alternative de la position des caméras sert pour l'apprentissage. En effet, les nouveaux angles de la tête n'ont pas besoin d'être appliqués en temps que commande. Il n'est pas utile (mais possible) de déplacer les caméras.

γ.– Apprentissage de $\mathrm{A_c} \rightarrow \Theta$

Le module $\mathrm{A_c} \rightarrow \Theta$ apprend la relation entre les informations de la tête et les angles du bras robotique. Pour que l'apprentissage puisse se faire, il faut qu'il existe une relation qui lie les entrées aux sorties. Comme cela a déjà été évoqué, il n'existe qu'un triplet d'angles α^c pour une position de l'effecteur dans l'espace 3D. D'un autre côté, il n'existe qu'une manière d'atteindre ce même point de l'espace pour le robot trois axes, si et seulement si les angles sont contraints de manière à supprimer toutes les redondances. Pour plus d'informations sur le modèle inverse du robot, nous pouvons nous reporter aux livres de Lallemand et Zeghloul (1994) et Coiffet (1992).

Ce module apprend donc la mise en série du modèle direct de la tête et du modèle géométrique inverse du bras robotique.

Le module utilisé pour apprendre ce modèle est un module à 3D comprenant trois entrées et trois sorties. Le tableau 3.3 donne une idée des coefficients utilisés. Le réseau est de taille $12 \times 9 \times 12$ et l'apprentissage se fait sur 15000 exemples.

Les entrées de sélections sont les angles α^c_{vg}, α^c_t et la disparité qu'il y a entre les deux vergences caméras centrées $(\alpha^c_{vg} - \alpha^c_{vd})$. Ces entrées sont normalisées de manière à ce qu'elles aient la même importance dans le choix du neurone vainqueur. La figure 3.12 montre l'espace des entrées. Les points noirs sont un échantillon représentatif des entrées de selections. La carte de Kohonen de l'espace d'entrées y est aussi représentée.

Nous voyons sur cette figure que la répartition des entrées n'est pas homogène. C'est pourquoi plus de neurones sont utilisés pour représenter au mieux les zones où la densité des vecteurs d'entrées est la plus importante.

La courbe observée est typique d'un apprentissage correct. Durant les premières itérations, l'erreur de discrétisation diminue rapidement. Au bout de 5000 itérations,

Structure du réseau : $100 \times 1 \times 1$	
SOM–LLM Dimension des vecteurs : 1+1	
Apprentissage : 5000 cycles, évolution : exponentielle	
Taux d'apprentissage d'entrée	$\mu_e : 0.7 \searrow 0.001$
Taux d'apprentissage de sortie	$\mu_s : 0.4 \searrow 0.25$
Taux d'apprentissage des jacobiennes	$\mu_j : 0.5 \searrow 0.25$
Rayon de voisinage d'entrée	$\sigma_e : 5 \searrow 0.001$
Rayon de voisinage de sortie	$\sigma_s : 5 \searrow 0.001$
Rayon de voisinage des jacobiennes	$\sigma_j : 5.5 \searrow 0.2$

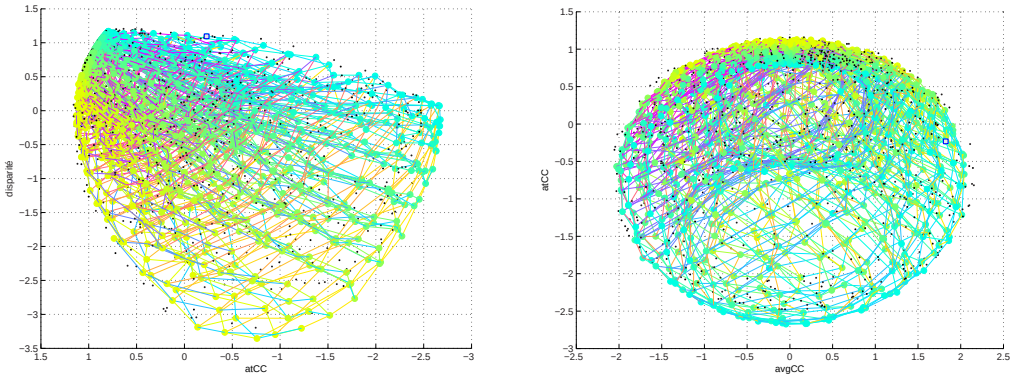
Tab. 3.1 *Coefficients d'apprentissage pour un réseau 1D à une entrée*

Structure du réseau : $100 \times 1 \times 1$	
SOM–LLM Dimension des vecteurs : 2+1	
Apprentissage : 5000 cycles, évolution : exponentielle	
Taux d'apprentissage d'entrée	$\mu_e : 0.7 \searrow 0.001$
Taux d'apprentissage de sortie	$\mu_s : 0.4 \searrow 0.25$
Taux d'apprentissage des jacobiennes	$\mu_j : 0.5 \searrow 0.25$
Rayon de voisinage d'entrée	$\sigma_e : 5 \searrow 0.001$
Rayon de voisinage de sortie	$\sigma_s : 5 \searrow 0.001$
Rayon de voisinage des jacobiennes	$\sigma_j : 5.5 \searrow 0.2$

Tab. 3.2 *Coefficients d'apprentissage pour un réseau 1D à deux entrées*

Structure du réseau : $12 \times 9 \times 12$	
SOM–LLM Dimension des vecteurs : 3+3	
Apprentissage : 15000 cycles, évolution : exponentielle	
Taux d'apprentissage d'entrée	$\mu_e : 0.7 \searrow 0.001$
Taux d'apprentissage de sortie	$\mu_s : 0.4 \searrow 0.25$
Taux d'apprentissage des jacobiennes	$\mu_j : 0.5 \searrow 0.25$
Rayon de voisinage d'entrée	$\sigma_e : 5 \searrow 0.001$
Rayon de voisinage de sortie	$\sigma_s : 5 \searrow 0.001$
Rayon de voisinage des jacobiennes	$\sigma_j : 5.5 \searrow 0.2$

Tab. 3.3 *Coefficients d'apprentissage pour un réseau 3D à trois entrées*



a) b)

Fig. 3.12 *Distribution des entrées de sélection pour le module*

Ces figures représentent deux vues de l'espace d'entrées du second module. Les points noirs représentent les entrées, la grille superposée représente la discrétisation de cet espace faite avec une carte de Kohonen.

la carte est relativement bien déployée. Des entrées de sélections supplémentaires permettent de modifier localement le placement des neurones (les coefficients d'apprentissage et de voisinage prennent de petites valeurs).

La figure 3.13 représente l'espace de sorties (les trois θ) et la carte de Kohonen associée. Les points verts sont les sorties désirées, ils sont disposés dans un demi-cube d'arrête π.

L'erreur d'estimation des θ est de l'ordre de 1 à 1.5 degrés en moyenne, et d'une dizaine de degrés au maximum. Les erreurs maximales se trouvent dans les zones de l'espace les moins bien représentées qui sont situées soit aux limites de l'espace, soit dans les zones désertées, soit aux limites entre une zone dense et une zone déserte. L'ordre de grandeur de l'erreur est étroitement lié au nombre de neurones utilisés pour discrétiser les espaces. Des erreurs d'estimation plus petites ont été obtenues lorsque nous avons utilisé notre décomposition sur une autre disposition de la scène robotique (Hermann *et al.*, 2003a). Un nombre de neurones équivalent a été utilisé pour un volume de travail plus petit. La décomposition que nous utilisons pour traiter notre tâche robotique est tout à fait générale et peut être appliquée à toutes sortes de configurations de l'environnement (position différente du robot, autres longueurs d'axes, tête robotique différente, etc.).

Maintenant que nous avons un système complet qui a été appris, il est intéressant d'utiliser nos modules en temps que contrôleurs dans une boucle de commandes et de quantifier ainsi l'erreur de positionnement. Le contrôleur prend en entrées les informations visuelles et détermine les angles du bras correspondant. Les sorties ainsi définies sont données au bras robotique en tant que commandes. En simulation, nous utilisons le modèle du bras défini par les équations (3.10), (3.11) et (3.12). Nous

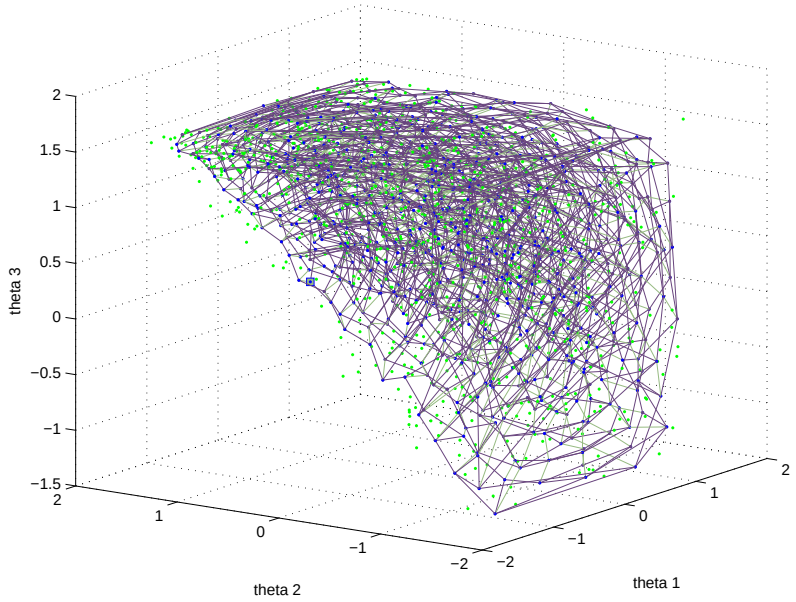


Fig. 3.13 *Discrétisation de l'espace de sortie*

La figure montre la carte de sorties du réseau de neurones. Les points verts représentent les sorties désirées (les θ sont exprimés en radians).

pouvons alors comparer les distances séparant l'effecteur observé dans les images avant estimation et l'effecteur positionné à la suite de commandes fournies par le contrôleur neuronal. L'erreur de positionnement moyenne dans l'espace 3D est de l'ordre de 3–4 millimètres. Les erreurs de positionnements maximales sont de l'ordre de quelques centimètres.

3.6 Le flux inverse

L'apprentissage que nous venons d'effectuer donne des résultats tout à fait satisfaisants. Il faut cependant garder à l'esprit que l'apprentissage tel qu'il a été mené n'est possible que dans l'environnement de simulation. En effet, les angles de la tête qui centrent l'effecteur dans les deux images, utiles à l'apprentissage du premier bloc, ne sont ni connus, ni mesurables en situation réelle. Il faut alors mettre en œuvre des structures d'aide à l'apprentissage.

Une solution, présentée par Buessler (1999), consiste à introduire un flux inverse pour permettre l'apprentissage. Ce flux peut être introduit de manières différentes : soit comme une information rétropropagée de la sortie vers les entrées (Bottou et Gallinari, 1991), soit en introduisant de nouveaux modules neuronaux. C'est cette seconde solution qui fait l'objet de ce travail.

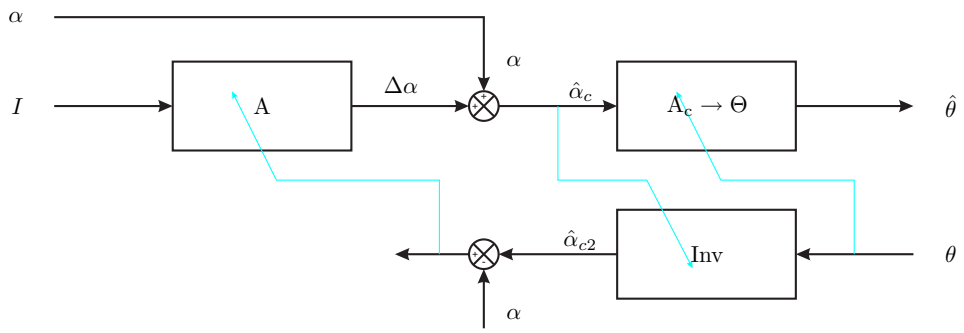


Fig. 3.14 *Architecture modulaire avec flux inverse*

Sur cette figure, nous retrouvons la décomposition de la tâche, ainsi que le bloc inverse et les flux d'apprentissage (en bleu). Ceux-ci indiquent la sortie désirée.

L'architecture à deux modules de la figure 3.9 va bénéficier d'un nouveau module qui va lui permettre l'apprentissage du module A. La nouvelle architecture peut être vue sur la figure 3.14.

Le nouveau module Inv va fournir le modèle d'apprentissage au module A. De la même manière, la sortie du module A va servir de modèle d'apprentissage au module Inv. L'apprentissage du module $A_c \rightarrow \Theta$ reste quant à lui inchangé.

Les deux modules A et Inv sont adaptés simultanément et tendent vers une même représentation de la variable interne, ici les α centrés, α^c.

La principale difficulté est de faire converger les sorties des deux modules vers une grandeur représentative des α^c. Mais avant toute chose, il faut s'assurer que le module Inv puisse apprendre. C'est-à-dire qu'il faut que le module $A_c \rightarrow \Theta$ soit inversible et qu'il n'y a pas, bien sûr, de redondance possible.

α.– Apprentissage de Inv

Ce module va apprendre le modèle inverse du module $A_c \rightarrow \Theta$. Ces entrées sont les trois angles θ. Ce sont en fait les sorties désirées servant à l'apprentissage du module $A_c \rightarrow \Theta$. Les sorties de ce module seront les angles α^c. L'espace intermédiaire entre celui du bras et celui de la tête est l'espace 3D. Le passage des θ aux coordonnées 3D de l'effecteur est bijectif. La relation est donnée par les équations (3.10), (3.11) et (3.12). La relation qui permet de passer des coordonnées 3D aux angles de la tête est aussi bijective, si les angles sont contraints de telle sorte que le placement des caméras donne un motif unique dans les images. Cette contrainte est bien entendu la même que celle appliquée au module A. Le module à entraîner apprend le modèle géométrique direct du robot trois axes en série avec le modèle inverse de la tête robotique.

Les paramètres d'apprentissage du module à trois dimensions sont les mêmes que ceux du module dont il faut apprendre l'inverse. Ils sont récapitulés dans la table 3.3.

L'apprentissage d'un tel module, seul, ne pose pas de problèmes particuliers. En effet, la relation entre les entrées et les sorties est bijective. De plus le réseau est de taille réduite et de dimension n'excédant pas trois. Les résultats obtenus vont en ce sens. Nous obtenons des erreurs d'estimation des angles de la tête α^c de l'ordre du dixième de degré.

β.– Apprentissage simultané de A et Inv

Les deux modules A (en fait $\alpha + \Delta\alpha$) et Inv estiment chacun les mêmes grandeurs, les α^c. Chacun des deux modules apprend respectivement une fonction f_1, qui utilise les informations visuelles, et f_2, qui utilise les angles du bras robotique, et fait une estimation de la sortie, respectivement $\hat{z_1}$ et $\hat{z_2}$. L'objectif est de réduire la différence entre les deux estimations $\hat{z_i}$.

Le schéma de décomposition que nous avons choisi est tel que la sortie estimée $\hat{z_1}$ est la sortie du premier module, à laquelle nous ajoutons les valeurs des différents angles α. Cela va contribuer à contraindre la convergence vers une représentation des α^c.

L'apprentissage simultané s'effectue de la manière suivante :

1. Des entrées de sélections sont présentées à chacun des deux modules. Ces entrées doivent être synchronisées. Les modules fournissent chacun leur réponse, respectivement $\hat{z_2}$ et $\hat{z_1}$.
2. Les exemples d'apprentissage sont calculés en fonction des équations suivantes :

$$\begin{aligned} \epsilon_z &= \hat{z_2} - \hat{z_1} \quad , \\ w^*_{s1} &= (\hat{z_1} + \epsilon_z - p_t) \quad , \\ w^*_{s2} &= \hat{z_2} - \epsilon_t \quad , \end{aligned}$$

3. Les exemples d'apprentissage sont présentés aux deux modules. Les poids de chacun de ceux-ci sont adaptés de manière classique.

La variable p_t représente une estimation de la moyenne des réponses du premier réseau. L'évolution de cette variable est fonction des réponses successives $\hat{z_1}$:

$$p_{t+1} = p_t + \gamma_p \left(\hat{z_1}(t) - p_t\right) \quad . \tag{3.23}$$

γ_p est un coefficient d'adaptation variable dans le temps. L'évolution des variables p_t, intervenant dans chacune des trois estimations des α^c, est montrée sur la figure 3.15. Le coefficient γ_p influe sur la manière dont évolue p_t. Ce coefficient doit être

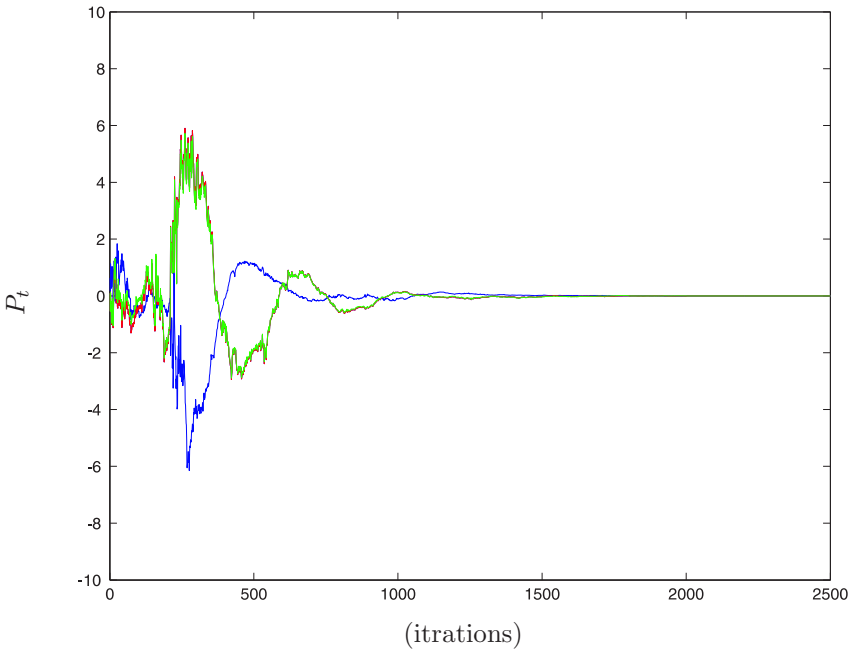


Fig. 3.15 *Exemple de convergence du coefficient P_t*

En abscisse le nombre de cycles, en ordonnée les valeurs de P_t.

assez grand pour prendre en compte les variations et les corriger. Sans cela, la boucle de correction risque d'être instable. En revanche, en fin d'apprentissage le coefficient doit être petit afin d'augmenter la fenêtre temporelle d'intégration et d'éviter le bruit d'apprentissage. Dans nos apprentissages, nous désactiverons totalement la régulation après quelques centaines de cycles d'apprentissage.

L'apprentissage des trois angles α^c se déroulent de la même manière. C'est pourquoi nous n'allons présenter que les résultats de obtenus pour l'estimation de α^c_{vg}. Ce choix est arbitraire. La figure 3.16 montre la sortie désirée α^c_{vg} en fonction de la sortie calculée $\hat{\alpha}^c_{vg}$. Nous constatons que les deux signaux sont très proches.

Les trois grandeurs estimées par les modules A et Inv vont servir d'entrées au module $\mathrm{A_c} \rightarrow \Theta$.

γ.– Apprentissage complet

L'apprentissage complet consiste en l'utilisation des estimations faites par l'apprentissage bidirectionnel comme entrées du module chargé de l'estimation des angles θ_1 à θ_3.

Dans ces apprentissages, il faut rester attentif au fait que les premières estimations issues de l'apprentissage bidirectionnel sont relativement éloignées des valeurs vers lesquelles elle devraient tendre. Les premières entrées du second module du flux direct sont donc mauvaises (au sens où elle ne correspondent pas aux sorties désirées

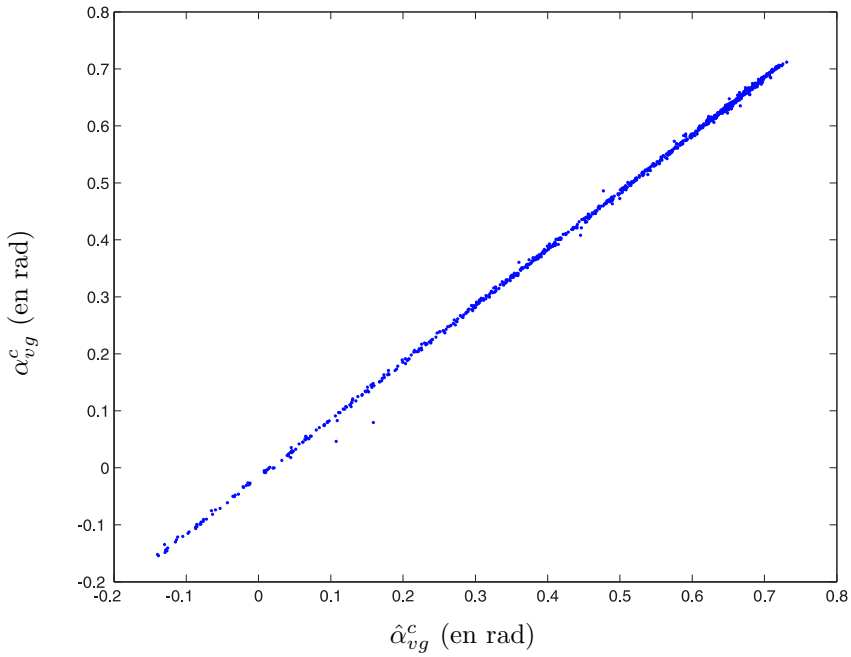


Fig. 3.16
$\alpha^c_{vg} = f(\hat{\alpha}^c_{vg})$

Cette courbe représente la sortie désirée et fonction de la sortie calculée. Les valeurs sont exprimées en radians.

présentées en même temps pour l'apprentissage). Par conséquent, les coefficients d'apprentissage de ce second module doivent rester forts assez longtemps pour qu'ils soient toujours influents dans l'algorithme d'apprentissage une fois que les sorties du premier module tendent vers des valeurs représentatives des angles α^c. Il reste tout à fait envisageable de différer de quelques cycles le début de l'apprentissage de ce second module. La méthode retenue n'influe pas sur la qualité des résultats, mais sur la facilité de mise en œuvre.

Les erreurs d'estimations des angles θ sont légèrement plus grandes dans cet apprentissage, par rapport à un apprentissage prenant les véritables valeurs des angles de la tête, qui permettent de centrer la cible dans les images. Ceci s'explique par le fait que dans le cas de l'apprentissage complet, la première étape qui est l'estimation des valeurs α^c introduit déjà une erreur. Les entrées du second module de l'architecture ne sont donc pas optimales. Il faut tout de même se souvenir que nous comparons ici deux estimations des angles θ, partant dans un premier temps des α^c et dans un second temps des informations images. Ce second modèle à apprendre est bien plus complexe. Bien entendu ces résultats sont à relativiser avec la taille de l'espace d'action du bras robotique ainsi qu'avec la taille des réseaux composant chacun des modules. Un exemple avec un espace de travail du robot restreint est présenté dans Hermann *et al.* (2003b).

3.7 Une décomposition alternative

Comme nous l'avons déjà évoqué un peu plus tôt, il n'existe pas une décomposition unique pour un problème donné. La décomposition que nous avons utilisée jusqu'à présent repose sur une modification de la position de la tête robotique. Les contraintes utilisées étaient un axe panoramique nul et des vergences telles que l'effecteur se projette au centre des deux images.

La décomposition que nous proposons maintenant est toujours basée sur une configuration alternative de la position de la tête qui centre la cible dans les deux images. Nous utilisons toujours cette configuration particulière de la tête, puisse qu'elle permet de s'affranchir des coordonnées de la cible dans les images. Cependant il existe plusieurs positions de la tête pour que la cible soit centrée. Garder l'angle panoramique nul permet de passer de N possibilités de centrer à une seule. La nouvelle configuration est basée sur la vergence symétrique et utilise les quatre angles de la tête. De la même manière, cette configuration particulière permet de passer de N possibilités de centrer la cible dans les images à une seule, et permet par conséquent de réduire le nombre d'informations permettant de positionner les deux caméras. Dans ce cas nous avons :

$$\alpha_{vg} = \alpha_{vd} = \alpha_v \quad . \tag{3.24}$$

Utiliser des vergences symétriques revient à modifier les angles α_{vg}, α_{vd} et α_p. La figure 3.17 montre un point P observé avec une configuration de vergence symétrique à un instant t (en noir) et ce même point observé à instant $t+1$ (en bleu). Le premier module de la décomposition calcule les différences angulaires à appliquer aux angles initiaux pour passer d'une configuration à l'autre. C'est une configuration typique de suivi de cible.

Les angles β_1 et β_2 découlent directement des informations images, comme le montrent les équations (3.18) et (3.19). Ils permettent de déterminer la variation d'angle sur l'axe panoramique, $\Delta\alpha_p$. Il existe donc une relation :

$$\Delta\alpha_p = f\left(\alpha_{v_t}, \beta_1, \beta_2\right) \quad , \tag{3.25}$$

de même qu'il existe une relation :

$$\Delta\alpha_v = f\left(\alpha_{v_t}, \beta_1, \beta_2\right) \quad , \tag{3.26}$$

qui détermine la variation d'angle sur les axes de vergences, $\Delta\alpha_v$, entre les deux positions de la tête.

La variation sur l'angle de tilt pour centrer la cible dans les images est determinée de la même manière de ce qui a été présenté dans une section précédente.

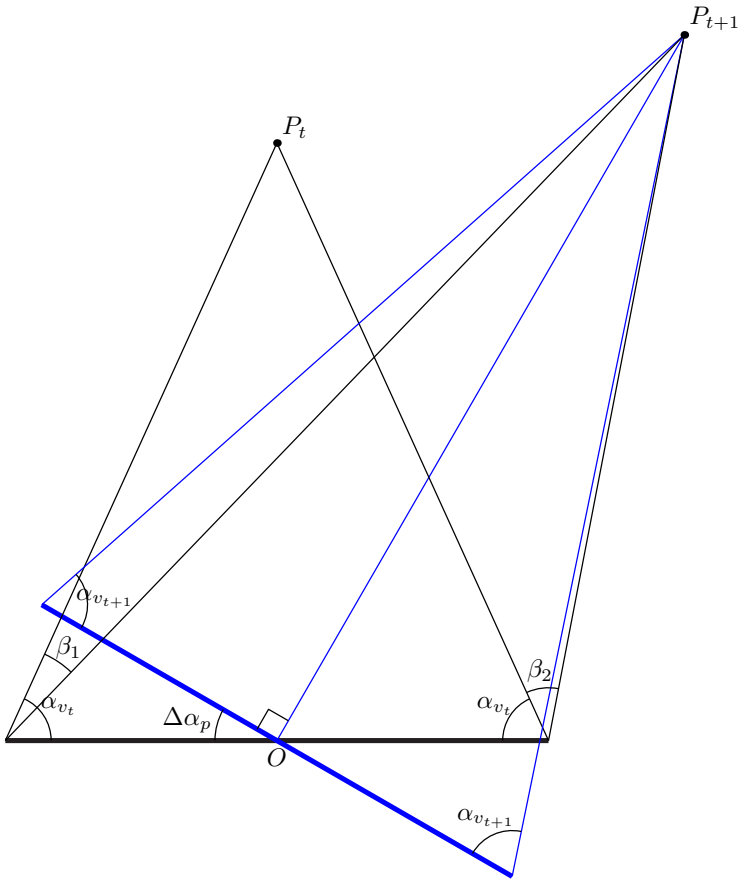


Fig. 3.17 *Vergence symétrique*

3.8 Discussion

Les réseaux de neurones constituent un outil important dans le domaine de l'estimation de fonctions. Nous avons montré dans un premier exemple, qu'une structure modulaire permet de les utiliser pour l'apprentissage de systèmes complexes. La complexité est définie, ici, par la taille du vecteur d'entrées.

Les cartes auto-organisatrices que nous utilisons font le lien entre un espace d'entrées et un espace de sorties. Plus l'espace d'entrées est complexe, plus la transformation est difficile à définir. Pour illustrer ce propos, nous avons utilisé un apprentissage classique, c'est-à-dire composé d'une carte unique, pour déterminer la configuration angulaire d'un robot trois axes en utilisant des informations issues d'une tête robotique. Les résultats ne sont pas probants, ils mettent en évidence les limites de ce type d'apprentissage.

Parmi les architectures modulaires présentées dans le chapitre 2, nous avons vu que les décompositions séquentielles permettent de traiter par parties, un vecteur d'entrées. Il faut pour cela qu'une décomposition puisse être dégagée.

Dans notre exemple, deux caméras observent l'effecteur d'un robot trois axes. Leurs orientations sont quelconques, ce qui signifie qu'une position de l'effecteur correspond à un nombre illimité de configurations de caméras. Par conséquent, la position de l'effecteur est définie par six variables (les coordonnées dans les images et les angles de la tête). La décomposition en modules nécessite une étude préalable du problème. De celle-ci la décomposition suivante a été dégagée : deux modules composent l'architecture. Un premier effectue un changement virtuel de la configuration de la tête, de manière à ce que l'effecteur se centre dans les deux images. De cette manière une position de l'effecteur correspond à une seule orientation des caméras. La position de l'effecteur peut alors être déterminée par les trois angles de la tête. Cette relation est apprise par le second module.

Il est intéressant de noter que la décomposition retenue n'est pas propre à la scène que nous avons utilisée. En effet, l'emploi d'une architecture modulaire ne modifie en rien les capacités d'adaptation des réseaux de neurones. La décomposition que nous proposons a été utilisée dans deux contextes différents. Le premier, celui utilisé dans ce chapitre, et un second, utilisé dans Hermann *et al.* (2003b,a). Dans ces deux cas, les robots sont différents (d'un point de vue longueurs des éléments et et d'un point de vu ancrage), les modèles de têtes sont différents, les positions relatives de la tête par rapport au robot sont différentes et enfin les espaces de travail des robots sont différents.

Les architectures modulaires de type séquentielles, comme celles que nous utilisons, nécessitent l'emploi de structures d'aide à l'apprentissage. Ces structures sont indispensables dans le cas où les grandeurs intermédiaires (entre deux modules) ne sont pas connues, ou pas mesurables. L'introduction d'un flux inverse permet de faire une estimation de ces grandeurs et rend par conséquent, l'apprentissage possible. L'architecture modulaire devient alors bidirectionnelle.

3.9 Conclusion

Dans ce chapitre, nous avons commencé par présenter l'environnement robotique qui sert de scène à nos applications.

L'exemple que nous avons traité a pour but de montrer qu'il est possible d'apprendre des fonctions complexes avec des réseaux de neurones, si ceux-ci sont organisés dans une structure modulaire.

Après avoir validé la décomposition, en apprenant la tâche module par module, nous avons intégré un flux inverse à l'architecture. Son rôle est de fournir des exemples d'apprentissages aux modules qui estiment des grandeurs intermédiaires.

Pour finir, nous avons discuté des bénéfices que nous avons à utiliser des architectures modulaires pour l'apprentissage de fonctions complexes. Afin de mettre en avant les avantages à utiliser de telles structures, nous avons appris en parallèle la même tâche robotique, avec un réseau de neurones unique.

Dans le chapitre suivant, nous allons utiliser des architectures bidirectionnelles pour estimer l'orientation de l'effecteur d'un robot quatre axes. Cette application va nous permettre de développer une architecture modulaire plus complexe mettant en œuvre un nombre de modules plus important.

4

Application au robot quatre axes

Sommaire

La modularité permet d'utiliser des réseaux de neurones pour apprendre des fonctions complexes. Un premier exemple a été développé dans le précédant chapitre, un second va être traité dans celui-ci. Il s'agit de déterminer l'orientation de l'effecteur d'un robot quatre axes, en utilisant les informations issues de la tête robotique. Cette tâche est bien plus complexe que la précédente puisqu'elle fait intervenir de nombreuses variables, telles que la projection de l'effecteur et du poignet dans les images, ou encore les orientations des caméras. Cette application est un travail préliminaire pour l'asservissement d'un bras robotique à une position de l'espace avec une orientation donnée.

Dans cette application, deux éléments du bras robotique sont observés : l'effecteur (extrémité du quatrième axe) et le poignet (extrémité du troisième axe). Nous nous intéressons dans cette partie à la détermination de l'orientation du dernier élément, et non pas à l'estimation de tous les angles du robot, puisque le cas des trois premiers axes trouve déjà une solution dans le chapitre 3.

La décomposition que nous allons proposer repose sur une analyse géométrique du problème. Chacune des variables intermédiaires représente une grandeur réelle. Il faudra, pour les estimer, mettre en œuvre plusieurs modules inverses.

Cette application constitue un deuxième exemple pour montrer que les réseaux de neurones peuvent être utilisés pour l'estimation de fonctions complexes, s'ils sont intégrés dans des architectures adaptées. Par ailleurs, cette application met en évidence les contraintes induites par l'emploi d'une architecture bidirectionnelle et l'emploi d'un nombre plus important de modules.

Après avoir présenté le robot quatre axes de l'application et la décomposition retenue, nous déterminerons les modèles de chaque module. Ceux-ci vont nous permettre de valider notre approche. Nous présenterons ensuite le schéma d'apprentissage utilisé puis les résultats obtenus.

4.1 Définition du robot 4 axes

Le robot que nous allons utiliser dans cette application est un robot quatre axes. Comme pour le robot trois axes que nous avons utilisé dans le chapitre 3, il correspond au robot de notre plate-forme expérimentale duquel nous n'utilisons que les quatre premiers axes.

L'angle θ_4, associé au dernier élément du bras, est défini par rapport au plan horizontal. Il se situe dans le même plan que les angles θ_2 et θ_3. Une vue schématique du bras robotique utilisé dans ce chapitre est montrée sur la figure 4.1. Une telle configuration permet à l'effecteur d'atteindre une position 3D avec une orientation.

Le modèle de ce robot est défini par les équations suivantes :

$$x = (\mathrm{l}_2 \cos\theta_2 + \mathrm{l}_3 \cos\theta_3 + \mathrm{l}_4 \cos\theta_4) \cos\theta_1 + \mathrm{x_b} \ , \tag{4.1}$$

$$y = (\mathrm{l}_2 \cos\theta_2 + \mathrm{l}_3 \cos\theta_3 + \mathrm{l}_4 \cos\theta_4) \sin\theta_1 + \mathrm{y_b} \ , \tag{4.2}$$

$$z = \mathrm{l}_2 \sin\theta_2 + \mathrm{l}_3 \sin\theta_3 + \mathrm{l}_4 \sin\theta_4 - \mathrm{l}_1 + \mathrm{z_b} \ . \tag{4.3}$$

Les longueurs l_1, l_2, l_3 et l_4, de chacun des membres du bras, valent respectivement 680 mm, 400 mm, 480 mm et 160 mm. Les limitations angulaires pour les trois premiers angles restent inchangées par rapport à la simulation précédente. L'angle θ_4 pourra prendre des valeurs allant de $-\pi$ à π.

4.2 Décomposition de la tâche retenue

Une possibilité pour déterminer θ_4 est de considérer la projection du dernier élément du bras robotique sur le plan horizontal (nous appelons *main* cet élément). Si l_4 est la longueur du quatrième segment et l'_4 sa longueur projetée, alors l'angle θ_4 vaut :

$$\theta_4 = \cos^{-1}\left(\frac{l'_4}{\mathrm{l}_4}\right) \ . \tag{4.4}$$

La main est définie par ses deux extrémités qui sont le poignet P et l'effecteur E. C'est à partir de l'observation de ces deux éléments qu'il est possible de déterminer la projection de la main. Les informations dont nous disposons sont les coordonnées du poignet et de l'effecteur dans chacune des images ainsi que les positions des deux caméras. Ces positions sont déterminées par l'angle panoramique α_p, l'angle azimutal α_t et les deux angles de vergences α_{vg} et α_{vd}, respectivement pour la caméra gauche et la caméra droite.

Dans le cas particulier où les caméras sont centrées sur le poignet, la projection de celui-ci sur le plan horizontal est relativement simple à déterminer. En effet, elle ne dépend que des angles α_v et α_t.

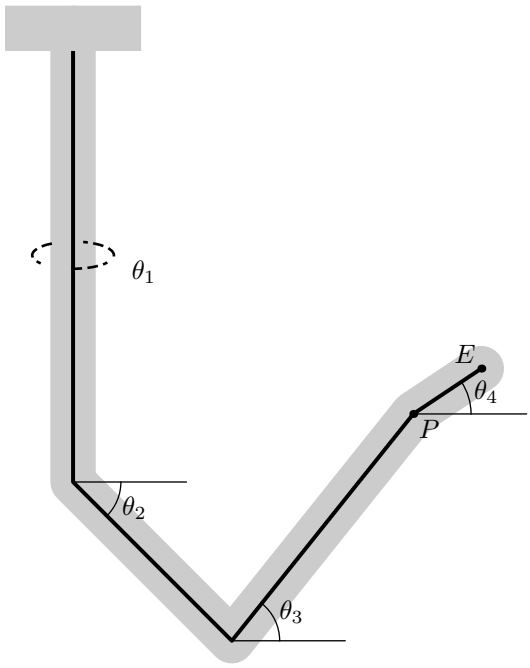


Fig. 4.1
Robot quatre axes

Vue schématique du robot quatre axes. Tous les angles sont définis par rapport à l'horizontale. Le point E représente l'effecteur.

Considérons le plan $\mathcal{F} \in \mathbb{R}^2$ formé par les deux points focaux des deux caméras et par le poignet. L'origine du repère utilisé est le point focal gauche F_g. Les deux vergences α_{vg} et α_{vd}, ou plus simplement α_v dans le cas de vergences symétriques, permettent de déterminer les coordonnées du poignet dans cet espace.

Les coordonnées du poignet sur le plan horizontal $\mathcal{H} \in \mathbb{R}^2$ peuvent être facilement obtenues. Dans le cas de la vergence symétrique, la projection suivant l'axe des abscisses est invariante et celle suivant l'axe des ordonnées ne dépend plus que de α_t.

L'idée que nous avons retenue utilise ce principe, qui est la détermination des coordonnées du poignet et de l'effecteur dans le plan $\mathcal{F}$ puis sur le plan horizontal $\mathcal{H}$. Il suffira ensuite de calculer la distance entre les deux points projetés sur $\mathcal{H}$ et de déterminer ainsi l'_4. Les techniques neuronales que nous allons employer vont permettre de déterminer les quatre coordonnées projetées sur le plan $\mathcal{H}$. La dernière étape de calcul sera faite de manière analytique. La figure 4.2 montre le principe de la décomposition que nous avons retenue. Les deux blocs en couleur sont ceux qui vont faire l'objet d'un apprentissage.

La figure 4.3 montre les projections du poignet et de l'effecteur sur chacun des plans $\mathcal{F}$ et $\mathcal{H}$. La projection de l'effecteur sur le plan horizontal est fonction de cinq variables. Pour ne pas traiter les cinq informations simultanément, nous décomposons cette tâche en estimant une projection sur le plan $\mathcal{F}$. Pour estimer P_E, nous estimons préalablement P''_E et P'_E.

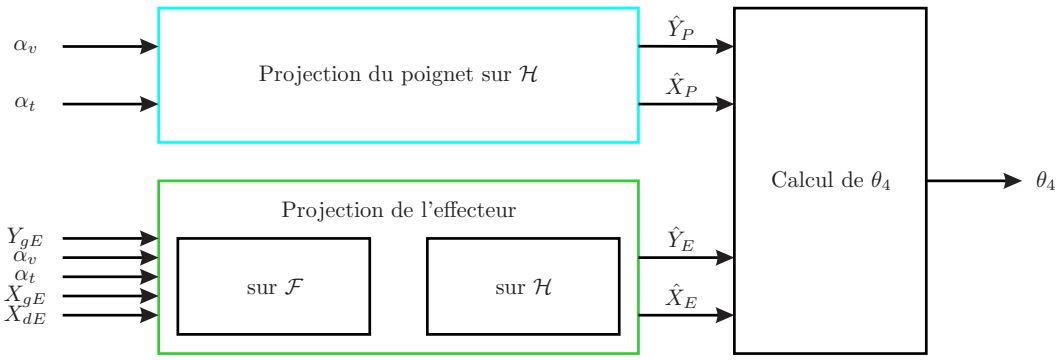


Fig. 4.2 *Principe de décomposition retenu*

Cette figure montre de quelle manière nous allons déterminer θ_4 en utilisant les informations images. Des projections successives permettent de déterminer les coordonnées du poignet et de l'effecteur sur le plan $\mathcal{H}$. Elles seront le résultat de techniques neuromimétiques.

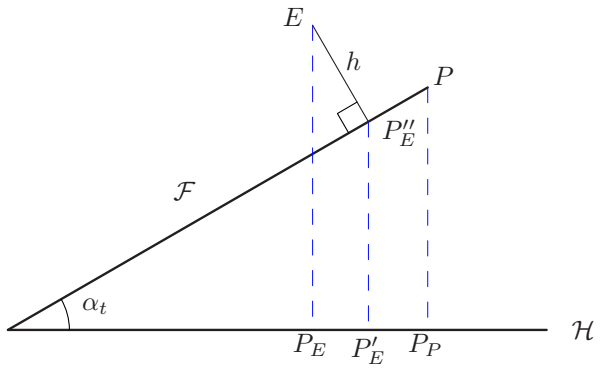


Fig. 4.3 *Les plans de projections*

Sur cette figure, nous pouvons voir une coupe des deux plans $\mathcal{H}$ et $\mathcal{F}$, ainsi que le poignet et l'effecteur, et leurs différentes projections.

Nous allons dans la section suivante déterminer les relations entre les grandeurs d'entrées dont nous disposons et les variables à estimer, c'est-à-dire les projections du poignet et de l'effecteur sur les différents plans. Ce sont ces relations qui vont nous fournir les modèles d'apprentissage et qui vont permettre de valider notre approche neuromimétique. Les développements vont être menés pour une tête robotique qui centre le poignet en utilisant des vergences symétriques.

4.3 Détermination des modèles

L'approche que nous avons retenue pour l'estimation de θ_4 repose sur la géométrie du système. Nous allons dans un premier temps déterminer les coordonnées du poignet

sur le plan $\mathcal{F}$ puis sur le plan $\mathcal{H}$. Dans un deuxième temps nous ferons les mêmes développement pour l'effecteur.

α.– Les grandeurs utilisées

La tête robotique observe le bras robotique caractérisé par deux éléments qui sont le poignet et l'effecteur. Nous faisons en sorte que le poignet soit centré dans les images et que les deux angles de vergence soient égaux. Cette configuration particulière peut être obtenue de diverses manières, soit par calcul ou par apprentissage.

Les informations dont nous disposons sont les suivantes :

- Les angles de la tête : α_p pour l'angle panoramique, α_v pour les vergences droite et gauche, α_t pour l'angle azimutal.
- Les grandeurs mesurées dans l'image gauche et droite concernant le poignet : I_x^p pour la coordonnée en x, I_y^p pour la coordonnée en y.
- Les grandeurs mesurées dans l'image gauche concernant l'effecteur : I_{xg}^e pour la coordonnée en x, I_{yg}^e pour la coordonnée en y.
- Les grandeurs mesurées dans l'image droite concernant l'effecteur : I_{xd}^e pour la coordonnée en x, I_{yd}^e pour la coordonnée en y.

Les grandeurs calculées sont définies ainsi :

- $P'_P = \{X'_P, Y'_P\}$ les coordonnées du poignet P sur le plan $\mathcal{F}$.
- $P_P = \{X_P, Y_P\}$ les coordonnées du poignet P sur le plan $\mathcal{H}$.
- $P''_E = \{X''_E, Y''_E\}$ les coordonnées de l'effecteur E sur le plan $\mathcal{F}$.
- $P'_E = \{X'_E, Y'_E\}$ les coordonnées de l'effecteur E sur le plan $\mathcal{H}$ avant correction.
- $P_E = \{X_E, Y_E\}$ les coordonnées de l'effecteur E sur le plan $\mathcal{H}$ après correction.

β.– Les projections du poignet

Sur le plan $\mathcal{F}$

La figure 4.4 représente le plan $\mathcal{F}$. $\mathrm{F_g}$ et $\mathrm{F_d}$ sont respectivement les points focaux gauche et droit des deux caméras. Les coordonnées P'_P dans le repère ainsi défini dépendent de la distance entre les deux points focaux B et de α_v.

La distance B dépend de α_v et de f la distance focale. Elle vaut :

$$B = 2\left(\mathrm{d} + \left(f \cos\left(\alpha_v\right)\right)\right) \quad , \tag{4.5}$$

avec d la demi-longueur de l'axe azimutal de la tête.

d vaut 125 mm sur la tête robotique de la plate-forme.

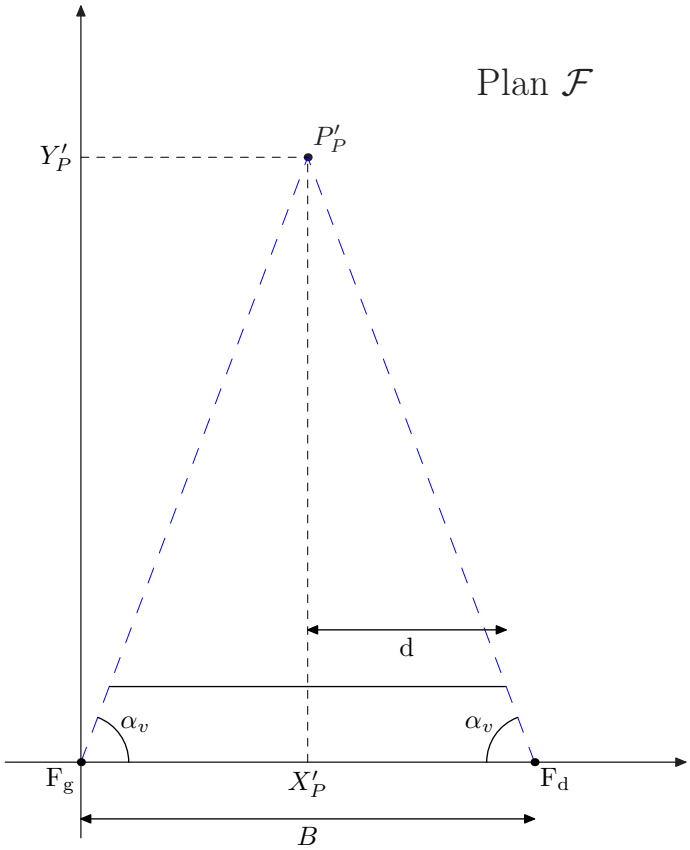


Fig. 4.4 *Représentation de P dans l'espace $\mathcal{F}$*

Le plan $\mathcal{F}$ est défini par les points focaux des deux caméras et le poignet.

Les coordonnées du poignet X'_P et Y'_P sont alors données par les équations (4.6) et (4.7) :

$$X'_P = \frac{B}{2} \ , \tag{4.6}$$

$$Y'_P = X'_P \tan(\alpha_v) \ . \tag{4.7}$$

Sur le plan $\mathcal{H}$

Il n'y a qu'une seule variable qui intervient pour le passage d'un plan à l'autre. C'est l'angle azimutal α_t représenté sur la figure 4.3.

Les deux espaces, ou plans de projections, sont définis de manière à ce que l'axe de tilt de la caméra soit parallèle à l'axe des abscisses. La conséquence de ceci est que les coordonnées en X des caméras restent inchangées lors du passage d'un plan à l'autre par rotation de l'axe de tilt. La coordonnée projetée en X du poignet sur le plan $\mathcal{H}$ est donc équivalente à celle du plan $\mathcal{F}$.

$$X_P = X'_P \ . \tag{4.8}$$

Soit le triangle défini par le poignet P, le centre de la tête et Y_P le projeté de P sur le plan horizontal (voir la figure 4.3), alors :

$$Y_P = Y'_P \cos(\alpha_t) \ . \tag{4.9}$$

γ.– Les projections de l'effecteur

Sur le plan $\mathcal{F}$

La géométrie de la tête ne suffit pas pour déterminer directement la projection de l'effecteur E sur le plan $\mathcal{F}$. Il faut prendre en compte les coordonnées image. Considérons les deux droites $(D_1) \in \mathcal{F}$ et $(D_2) \in \mathcal{F}$ (voir la figure 4.5). Ces deux droites se croisent en P''_E, les coordonnées projetées de E sur le plan $\mathcal{F}$. Ces coordonnées se déduisent des équations de chacune des droites.

Équation de (D_1) : L'équation de (D_1) est du type $y = \frac{\mathrm{a}}{\mathrm{b}}x$. La figure 4.6 a) est un zoom de la figure 4.5 et montre les diverses grandeurs qui entrent en jeu pour le calcul de (D_1). dx est la taille en X d'un pixel de la caméra soit 11μm.

Nous obtenons :

$$y = \frac{f \sin\alpha_v - I^e_{xg}\mathrm{dx}\cos\alpha_\mathrm{v}}{f\cos\alpha_v + I^e_{xg}\mathrm{dx}\sin\alpha_\mathrm{v}}x \ . \tag{4.10}$$

Équation de (D_2) : L'équation de (D_2) est du type $y = \frac{\mathrm{c}}{\mathrm{d}}x - \frac{\mathrm{c}}{\mathrm{d}}B$. La figure 4.6 b) est un zoom de la figure 4.5 au niveau du point focal droit. Cette zone nous intéresse pour déterminer l'équation de (D_2).

Nous obtenons :

$$y = \frac{f \sin\alpha_v + I^e_{xd}\mathrm{dx}\cos\alpha_\mathrm{v}}{-f\cos\alpha_v + I^e_{xd}\mathrm{dx}\sin\alpha_\mathrm{v}}x - B\frac{f \sin\alpha_v + I^e_{xd}\mathrm{dx}\cos\alpha_\mathrm{v}}{-f\cos\alpha_v + I^e_{xd}\mathrm{dx}\sin\alpha_\mathrm{v}} \ . \tag{4.11}$$

Intersection de (D_1) et de (D_2) : Les équations (4.10) et (4.11) permettent de définir l'intersection des deux droites et par conséquent les coordonnées projetées de l'effecteur sur le plan $\mathcal{F}$.

Nous obtenons :

$$X''_E = \frac{-B.b.c}{a.d - b.c} \ , \tag{4.12}$$

$$Y''_E = \frac{-B.a.c}{a.d - b.c} \ , \tag{4.13}$$

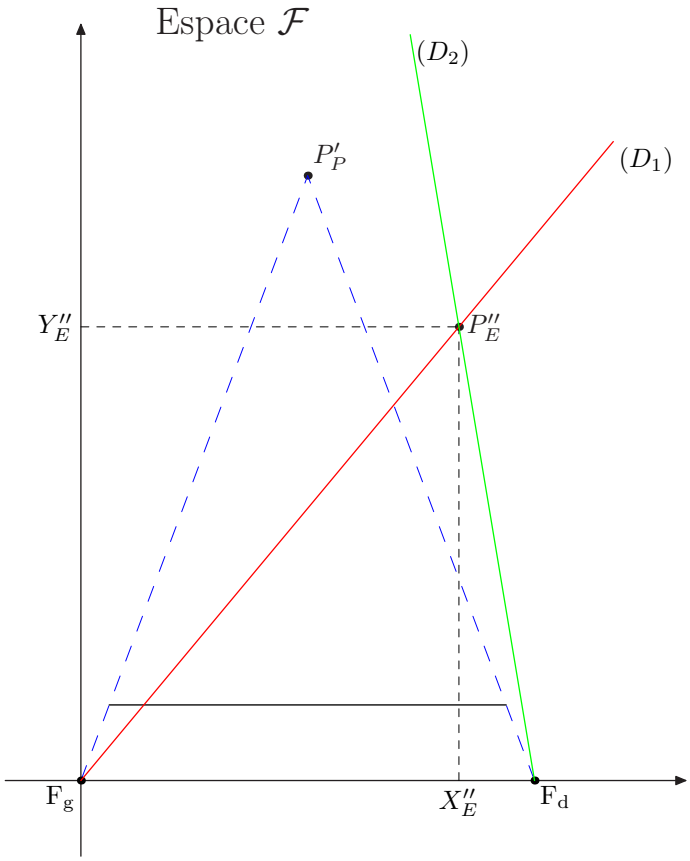
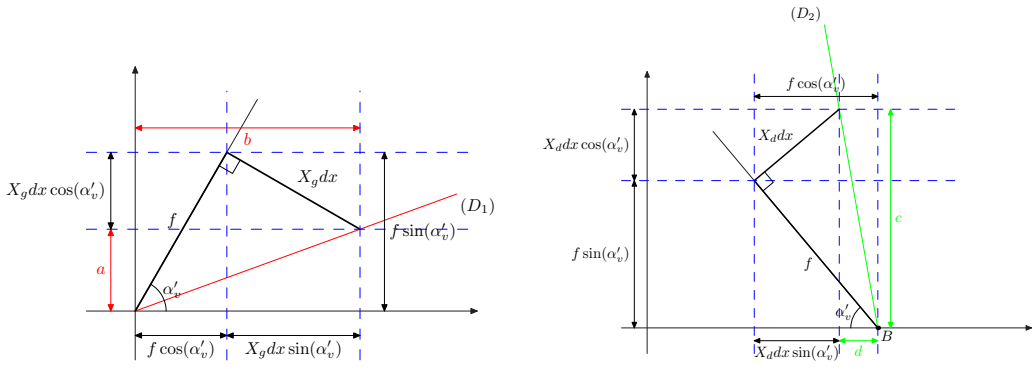


Fig. 4.5 *Représentation de P''_E dans l'espace $\mathcal{F}$*

Les coordonnées P''_E de l'effecteur projeté sur le plan $\mathcal{F}$ se calculent à partir des deux droites (D_1) et (D_2). Les équations de ces droites sont déterminées à partir des informations visuelles.



a) b)

Fig. 4.6 *Détermination de P''_E dans l'espace $\mathcal{F}$*

Nous utilisons les informations visuelles pour déterminer les équations des droites (D_1) et (D_2) (respectivement les figures a) et b)).

avec :

$$a = f\sin(\alpha_v) - I^e_{xg}\mathrm{dx}\cos(\alpha_\mathrm{v}) \ , \tag{4.14}$$
$$b = f\cos(\alpha_v) + I^e_{xg}\mathrm{dx}\sin(\alpha_\mathrm{v}) \ , \tag{4.15}$$
$$c = f\sin(\alpha_v) + I^e_{xd}\mathrm{dx}\cos(\alpha_\mathrm{v}) \ , \tag{4.16}$$
$$d = -f\cos(\alpha_v) + I^e_{xd}\mathrm{dx}\sin(\alpha_\mathrm{v}) \ . \tag{4.17}$$

Sur le plan $\mathcal{H}$

La projection de l'effecteur sur le plan $\mathcal{H}$ peut être considérée comme le projeté de l'effecteur sur le plan $\mathcal{F}$ projeté sur $\mathcal{H}$, auquel une correction est apportée (voir la figure 4.3). Cette correction n'a lieu que pour la coordonnée en Y. Les raisons pour lesquelles il y a une invariance suivant X sont les mêmes que celles exposées dans la partie $\beta.-$.

Nous avons alors :

$$X_E = X'_E = X''_E \ . \tag{4.18}$$

La projection Y'_E faite sur $\mathcal{H}$ de Y''_E ne dépend que de α_t. Nous obtenons :

$$Y'_E = Y''_E\cos(\alpha_t) \ . \tag{4.19}$$

La correction à apporter à Y'_E pour obtenir Y_E, la coordonnée en Y de l'effecteur sur le plan horizontal, est fonction de h, la distance qui sépare le plan $\mathcal{F}$ et le plan qui lui est parallèle et qui passe par l'effecteur. Pour calculer cette distance, l'information visuelle issue d'une caméra est suffisante. Nous choisissons arbitrairement la gauche (puisque F_g est l'origine de notre repère).

La figure 4.7 représente les différentes grandeurs utilisées pour calculer h.

Nous avons :

$$h = \frac{\sqrt{X''^2_E + Y''^2_E}I^e_{yg}}{f} \ . \tag{4.20}$$

La correction vaut alors $h\sin(\alpha_t)$. Nous obtenons :

$$Y_E = Y'_E - h\sin(\alpha_t) \ . \tag{4.21}$$

Les coordonnées projetées du poignet et de l'effecteur sont maintenant connues. Il est donc possible de déterminer l_4 et de calculer θ_4 en utilisant la formule (4.4).

$$l'_4 = \sqrt{(X_P - X_E)^2 + (Y_P - Y_E)^2} \ . \tag{4.22}$$

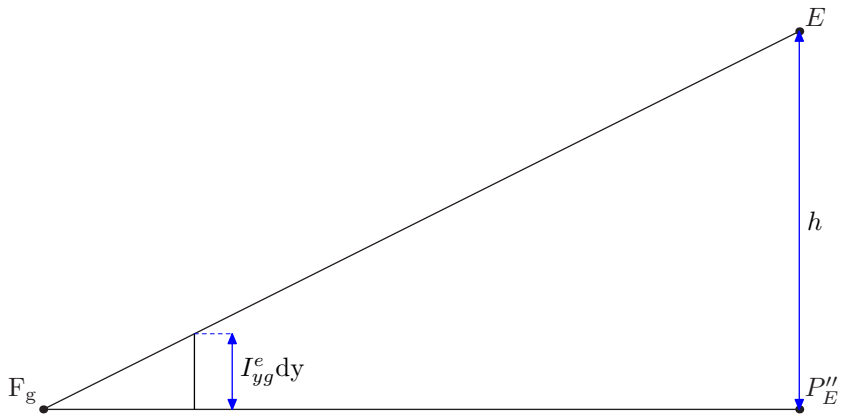


Fig. 4.7 *Calcul de la distance h*

La distance h se mesure entre le plan $\mathcal{F}$ et le plan parallèle à celui-ci, qui passe par l'effecteur. Cette distance permet de déterminer les coordonnées du projeté de E sur le plan horizontal $\mathcal{H}$.

Le développement géométrique que nous venons d'effectuer va nous permettre de dégager une décomposition modulaire pour organiser les réseaux neuronaux, de telle sorte à ce qu'ils puissent apprendre la relation qui lie les informations visuelles à l'orientation de la main du robot.

La décomposition présentée ici n'est pas la seule possible. En effet, une autre solution consiste en l'estimation de l'angle formé entre le plan contenant les axes du robot et un plan image. Cette méthode fait intervenir θ_1, le premier axe de rotation du bras.

La solution que nous avons présentée a été retenue parce qu'elle ne fait intervenir que les informations visuelles. Ainsi, elle peut s'appliquer à n'importe quel objet observé au travers des caméras. La condition reste bien sûr de connaître la taille de l'objet observé.

Dans ce qui va suivre, nous allons utiliser des réseaux de neurones pour faire l'estimation des quatre coordonnées. L'architecture adoptée est une composition de plusieurs réseaux. Les équations et les grandeurs que nous avons définies dans cette première partie du chapitre vont servir de modèle aux différents réseaux que nous allons entraîner. Nous utiliserons cette formulation théorique pour situer les réponses des réseaux de neurones qui auront appris et évaluer ainsi la qualité de l'apprentissage.

4.4 Schéma d'apprentissage proposé

Parmi tous les schémas d'apprentissage possibles pour réaliser la tâche que nous nous sommes fixée, celui que nous retenons est un schéma où chacun des blocs le composant fait l'estimation d'une grandeur physique (à laquelle nous pouvons donner

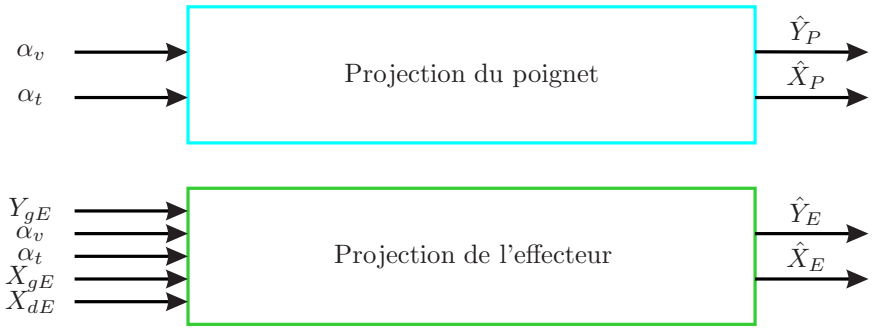


Fig. 4.8 *Premier niveau de décomposition*

Plusieurs niveaux de décomposition se dégagent de notre étude. Les traitements du poignet et de l'effecteur peuvent être effectués en parallèle puisqu'ils ne sont pas liés.

une signification) et dont le nombre d'entrées n'excède pas trois. Nous limitons les vecteurs d'entrées à trois éléments, pour avoir une bonne maîtrise de l'apprentissage et de la convergence des cartes auto-organisatrices qui composent chaque module. Ce schéma d'apprentissage reprend la décomposition que nous avons définie plus haut.

Plusieurs niveaux de décomposition se dégagent de notre étude. Le premier, trivial, sépare le traitement concernant le poignet et celui concernant l'effecteur. Ces deux traitements peuvent être effectués en parallèle puisque les estimations des projections du poignet et celles de l'effecteur ne sont pas liées. Ce premier niveau de décomposition est illustré sur la figure 4.8. Sur cette figure le terme « Entrées », générique, représente toutes les entrées disponibles et utilisables. Dans notre cas, les entrées sont les informations images (coordonnées du poignet et de l'effecteur dans les deux images) et les positions de la tête. Ces entrées sont disponibles, mais pas forcément utilisées par chacun des blocs. Les deux blocs peuvent être entraînés indépendamment ou simultanément.

Le second niveau de décomposition correspond aux découpages faits à l'intérieur de chacun des deux blocs vus précédemment. La nouvelle décomposition est série. Les sorties de certains modules servent d'entrées aux autres modules.

Le bloc que nous avons appelé « Projection du poignet » n'est composé que d'un seul module. Les projections du poignet ne dépendent que de α_v et α_t, en raison de la configuration particulière de la tête. Pour mémoire, les équations (4.8) et (4.9) montrent les relations à apprendre entre les entrées et les sorties de ce premier bloc. Ce module sera composé d'un SOM-LLM unique. La figure 4.9 montre ce bloc avec ses entrées et ses sorties. Le « ˆ » spécifie que la grandeur est estimée.

Les équations allant de (4.12) à (4.21) montrent que le bloc « Projection de l'effecteur » a cinq entrées : α_v et α_t qui sont les positions angulaires de la tête, I^e_{xg}, I^e_{xd} et I^e_{yg} qui sont des informations extraites des caméras.

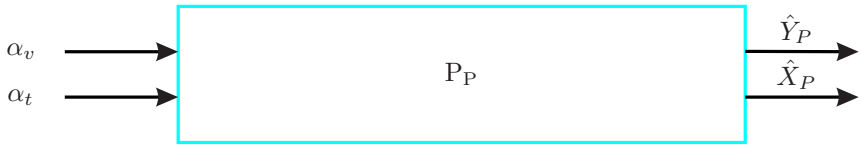


Fig. 4.9 *Module de projection du poignet*

Ce module permet de faire les estimations des coordonnées projetées du poignet. Les entrées sont les angles de vergence et de tilt.

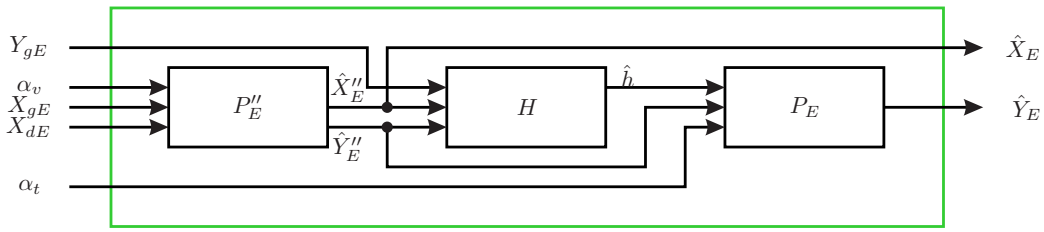


Fig. 4.10 *Module de projection de l'effecteur*

L'estimation de la projection de l'effecteur sur le plan horizontal nécessite trois modules dans le flux direct. Le premier procède à l'estimation des coordonnées projetées sur le plan $\mathcal{F}$, le second fait une estimation de h et le troisième fournit l'estimation des coordonnées de l'effecteur sur le plan $\mathcal{H}$.

Un module unique à cinq entrées ne convient pas, puisqu'il ne répond pas aux critères que nous nous sommes fixés, à savoir pas plus de trois entrées. La décomposition de ce bloc va donc suivre le schéma proposé en début de chapitre. Un premier module va calculer les estimations de la projection de l'effecteur sur le plan $\mathcal{F}$, qui ne dépend que de α_v, I^e_{xg} et de I^e_{xd}. Les sorties sont $\hat{X}''_E$ et $\hat{Y}''_E$. Les connaissances que nous avons du problème nous permettent de dire que $\hat{X}_E \equiv \hat{X}''_E$ dans le repère choisi.

Un second module va se charger de faire l'estimation de la projection de $\hat{Y}''_E$ sur le plan $\mathcal{H}$. Comme nous l'avons déjà vu, cette projection se fait en deux temps (projection de Y''_E et correction). L'estimation $\hat{Y}_E$ se fait avec $\hat{Y}''_E$, $\hat{X}''_E$, α_t et I^e_{yg} soit quatre entrées. Nous scindons donc ce deuxième module en deux modules : le premier fera l'estimation $\hat{h}$ de h et le second fera l'estimation $\hat{Y}_E$.

Pour résumer, l'estimation de la projection de l'effecteur sur le plan horizontal $\mathcal{H}$ se décompose en trois blocs, représentés sur la figure 4.10.

Une décomposition séquentielle, comme celle que nous venons de décrire, nécessite la mise en place d'une structure d'aide à l'apprentissage pour fournir des exemples d'apprentissage aux deux premiers blocs. Nous détaillerons les flux inverses dans la section suivante.

4.5 Résultats de l'apprentissage

Dans cette section, nous allons présenter les résultats d'apprentissage de chacun des blocs de l'architecture modulaire que nous avons présentée. Tous les blocs ont été entraînés simultanément.

α.– Le cas du poignet

Le bloc d'apprentissage de la projection du poignet est constitué d'un module unique. Un seul réseau de neurones de type SOM-LLM à deux dimensions est donc utilisé. La carte utilisée présente des paramètres équivalents de ceux présentés dans la table 4.1.

La carte d'entrées est de taille 10×11. Les vecteurs d'entrées sont $[\alpha_v,\alpha_t]$. La carte de sorties est de taille 10×11. Les sorties désirées sont $[X_P,Y_P]$. L'apprentissage est effectué sur la base de 5000 exemples.

La relation à apprendre est relativement simple et relève d'un apprentissage tout à fait classique. L'erreur de discrétisation de la carte d'entrées devient vite très faible. De même, les erreurs d'estimations décroissent très vite. Les figures 4.11 et 4.12 montrent les erreurs d'estimations au cours de l'apprentissage. La première courbe correspond à l'estimation de X_P et la seconde courbe correspond à l'estimation de Y_P.

En prolongeant quelque peu l'apprentissage, il est encore possible de diminuer les erreurs moyenne et maximale. L'erreur d'estimation est de l'ordre de 1% en moyenne, ce qui est tout à fait satisfaisant pour la taille du réseau choisi. Les courbes des figures 4.13 et 4.14 montrent les sorties désirées en fonction des sorties calculées. Les courbes obtenues sont proches de droites qui passeraient par l'origine avec un coefficient directeur de un.

β.– Le cas de l'effecteur

Nous allons dans cette section apprendre les coordonnées de la projection de l'effecteur sur le plan horizontal. Nous allons montrer l'apprentissage bloc par bloc en suivant la décomposition de la figure 4.15.

Taux d'apprentissage d'entrée	$\mu_e : 0.8 \searrow 0.002$
Taux d'apprentissage de sortie	$\mu_s : 0.8 \searrow 0.28$
Taux d'apprentissage des jacobiennes	$\mu_j : 0.5 \searrow 0.25$
Rayon de voisinage d'entrée	$\sigma_e : 5 \searrow 0.0003$
Rayon de voisinage de sortie	$\sigma_s : 8 \searrow 0.008$
Rayon de voisinage des jacobiennes	$\sigma_j : 5 \searrow 0.2$

Tab. 4.1 *Coefficient d'apprentissage pour un réseau 2D*

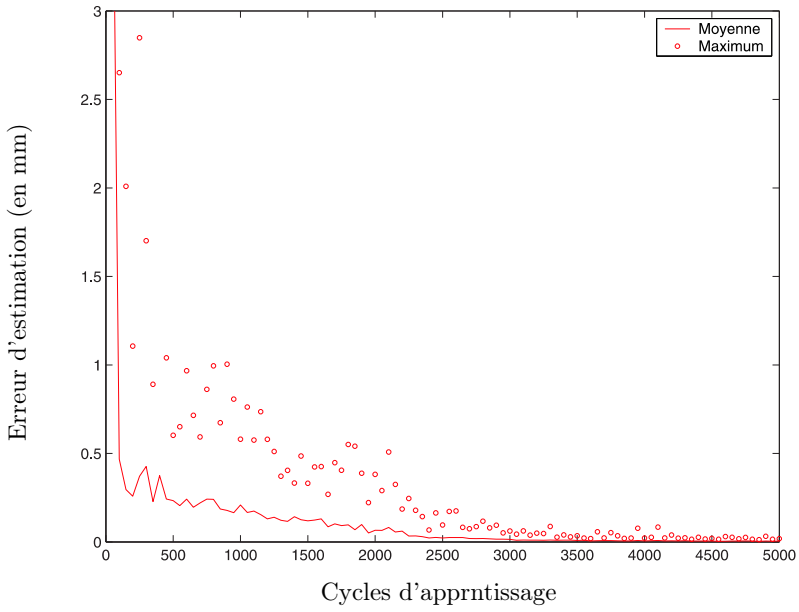


Fig. 4.11 *Erreur d'estimations au cours de l'apprentissage de X_P*

La courbe en trait continu montre l'erreur moyenne d'estimation et les erreurs maximum sont marquées avec des o. L'erreur est exprimée en millimètre (l'axe des ordonnées) en fonction des exemples d'apprentissage (l'axe des abscisses).

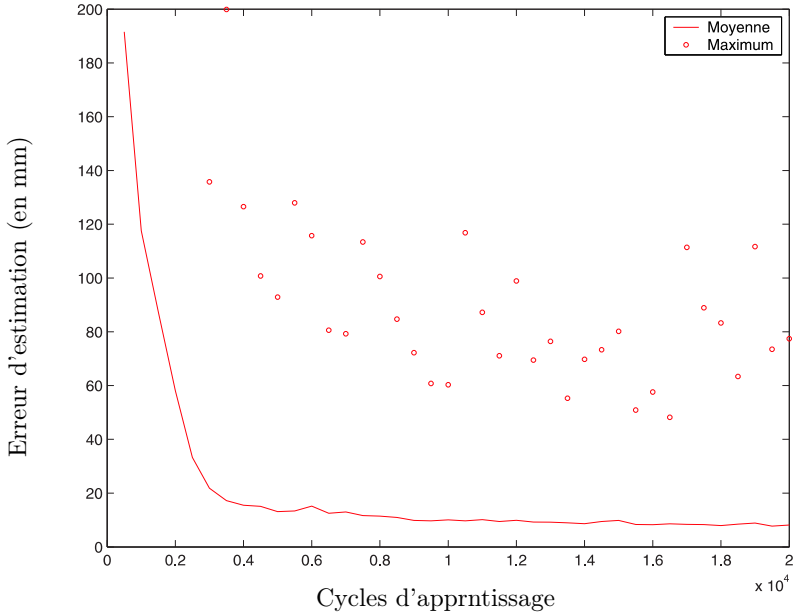


Fig. 4.12 *Erreur d'estimations au cours de l'apprentissage de Y_P*

La courbe en trait continu montre l'erreur moyenne d'estimation et les erreurs maximum sont marquées avec des o. L'erreur est exprimée en millimètre (l'axe des ordonnées) en fonction des exemples d'apprentissage (l'axe des abscisses).

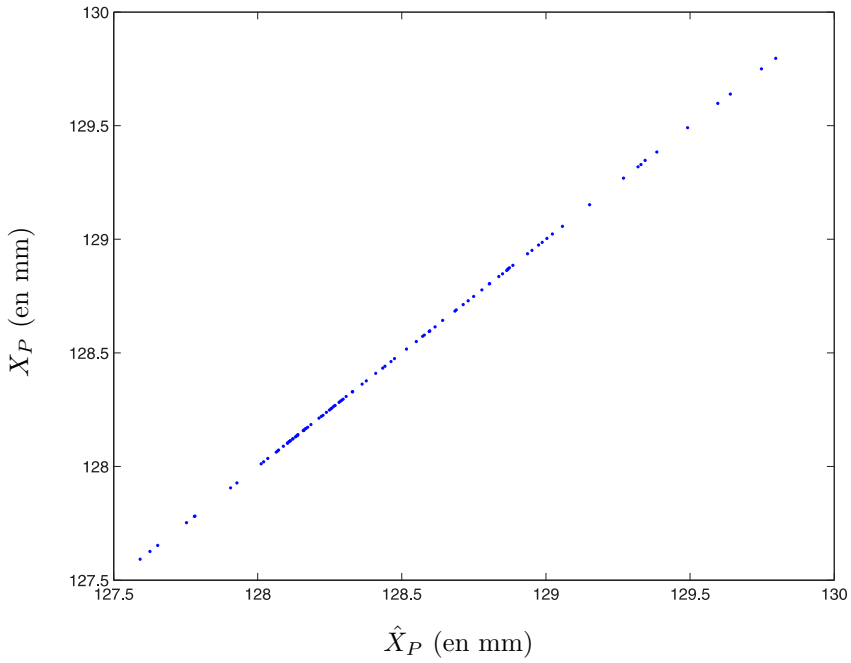


Fig. 4.13
Relation
$X_P = f(\hat{X}_P)$

Cette courbe représente la sortie désirée en fonction de la sortie calculée. Les valeurs sont exprimées en millimètre.

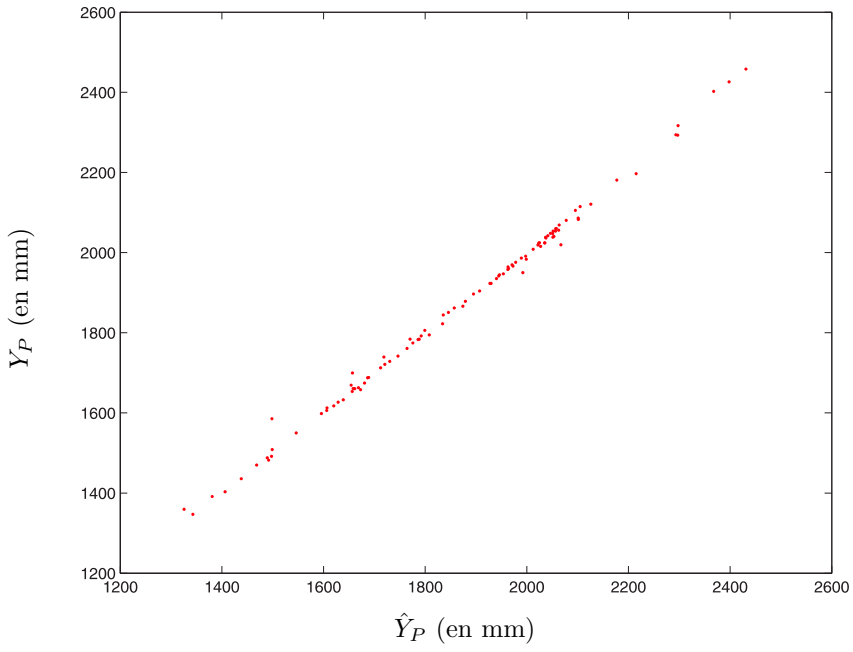


Fig. 4.14
Relation
$Y_P = f(\hat{Y}_P)$

Cette courbe représente la sortie désirées en fonction de la sortie calculée. Les valeurs sont exprimées en millimètre.

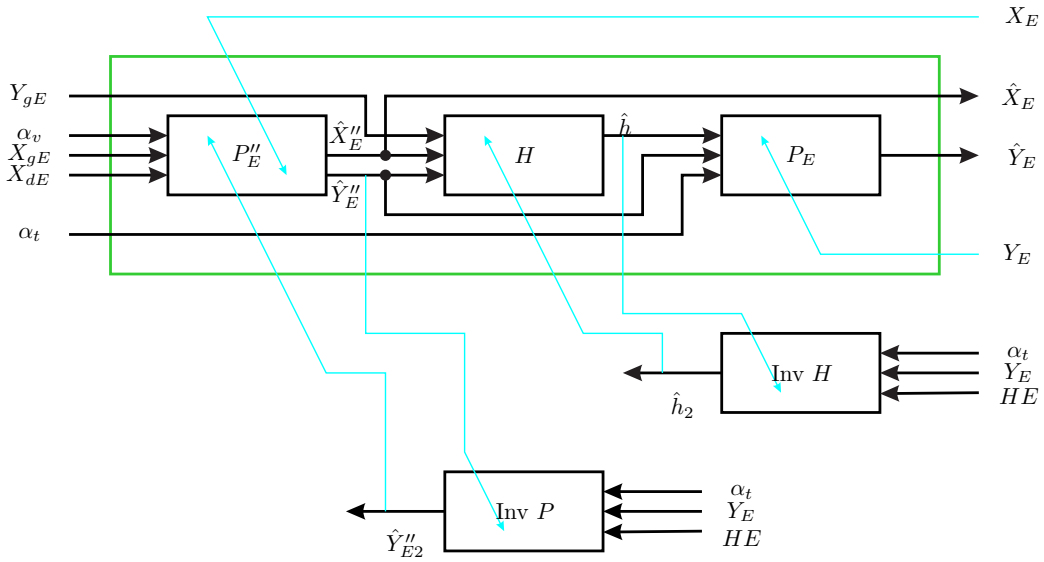


Fig. 4.15 *Architecture modulaire complète pour déterminer la projection de l'effecteur*

Cette figure présente l'architecture complète qui permet de déterminer les coordonnées projetées de l'effecteur sur le plan $\mathcal{H}$. En bleu sont représentés les flux d'apprentissage. Ils correspondent aux sorties désirées.

Apprentissage du premier bloc : projection sur $\mathcal{F}$

Ce premier bloc apprend les coordonnées projetées de l'effecteur sur le plan $\mathcal{F}$ des axes focaux. Le SOM–LLM utilisé pour cet apprentissage est à trois dimensions. La carte d'entrées est de taille $11 \times 10 \times 4$. Le vecteur d'entrées est composé de $[\alpha_v, I^e_{xg}, I^e_{xd}]$. Ces entrées doivent être normées pour avoir le même ordre de grandeur, et pour que chacune d'entre elles contribue avec la même importance à l'apprentissage. Les sorties désirées sont $[X''_E, Y''_E]$. Les valeurs des paramètres d'apprentissage sont présentées dans la table 4.2.

Comme X''_E ne subit pas de transformation au cours de la projection sur $\mathcal{H}$, nous pouvons prendre, de manière équivalente, le vecteur $[X_E, Y''_E]$ comme sorties désirées. L'intérêt de procéder de la sorte est que la grandeur X_E est accessible pendant l'apprentissage. Nous n'avons donc pas besoin de faire une estimation de X''_E.

En revanche pour apprendre la grandeur Y''_E, il faut créer des exemples d'apprentissage. Ces exemples sont des estimations faites par un bloc, Inv P, que nous ajoutons

Tab. 4.2 *Coefficient d'apprentissage pour un réseau 3D*

Taux d'apprentissage d'entrée	$\mu_e : 0.8 \searrow 0.002$
Taux d'apprentissage de sortie	$\mu_s : 0.8 \searrow 0.28$
Taux d'apprentissage des jacobiennes	$\mu_j : 0.5 \searrow 0.25$
Rayon de voisinage d'entrée	$\sigma_e : 5 \searrow 0.0003$
Rayon de voisinage de sortie	$\sigma_s : 8 \searrow 0.008$
Rayon de voisinage des jacobiennes	$\sigma_j : 5 \searrow 0.2$

à la structure, dans le flux inverse. Ce nouveau bloc fait une seconde estimation de Y''_E.

L'expérience montre que si les deux modules qui font l'estimation de Y''_E ont des caractéristiques similaires (taille et variation des coefficients), ils convergent plus facilement vers une solution acceptable.

Le module Inv P fait une seconde estimation de Y''_E. Les paramètres d'apprentissage de ce réseau sont les mêmes que ceux du module direct. Ce module prend ses entrées dans les espaces d'entrées et de sorties, elles sont α_t l'angle de tilt de la tête, Y_E la coordonnée projetée de l'effecteur sur $\mathcal{H}$ et HE, la hauteur de l'effecteur dans l'espace 3D (Z_E). Ces trois grandeurs permettent de déterminer Y''_E de la manière suivante :

$$Y''_E = Y_E + (HE - Y_E \tan(\alpha_t)) \tan(\alpha_t) \ . \tag{4.23}$$

L'apprentissage du module P''_E a ceci de particulier, c'est qu'il utilise des sorties désirées de deux natures différentes. Les premières sont mesurées (X_E), alors que les secondes sont issues d'une estimation ($\hat{Y}''_{E2}$). Les courbes 4.16 et 4.17 montrent les résultats d'un tel apprentissage.

Il est intéressant de comparer ces deux courbes. La première est très proche des courbes que nous avons obtenues pour l'apprentissage du module $\mathrm{P_P}$. Elle est significative d'un apprentissage classique. Les sorties calculées sont semblables aux sorties désirées. En revanche, sur la courbe issue de l'apprentissage bidirectionnel nous pouvons noter deux différences majeures :

1. Le facteur d'échelle entre les sorties désirées et calculées n'est pas de un. Comme nous pouvons le voir sur la figure 4.17, il est possible que ce facteur d'échelle soit négatif. La raison pour laquelle nous ce facteur d'échelle est différent, est que la convergence des deux modules vers une estimation représentative est contrainte. Les propriétés statistiques que nous souhaitons obtenir ne correspondent pas aux propriétés statistiques du signal désiré, ici, Y''_E (en termes de moyenne et de variance). Cependant les sorties estimées restent représentatives des sorties désirées.
2. La courbe représentant la sortie désirée en fonction de la sortie calculée n'est plus une droite, mais simplement une relation bijective dont les formes peuvent varier d'un apprentissage à l'autre.

L'allure des fonctions estimées lors d'un apprentissage bidirectionnel dépend en partie des contraintes appliquées aux signaux. Comme cela a déjà été mentionné, ces contraintes permettent de faire converger les algorithmes vers des solutions significatives. Elles permettent aussi de conditionner les valeurs estimées, pour qu'elles puissent être utilisées comme entrées des modules suivants. En effet, les entrées d'un module, qui participent à l'élection d'un neurone vainqueur, doivent être du même

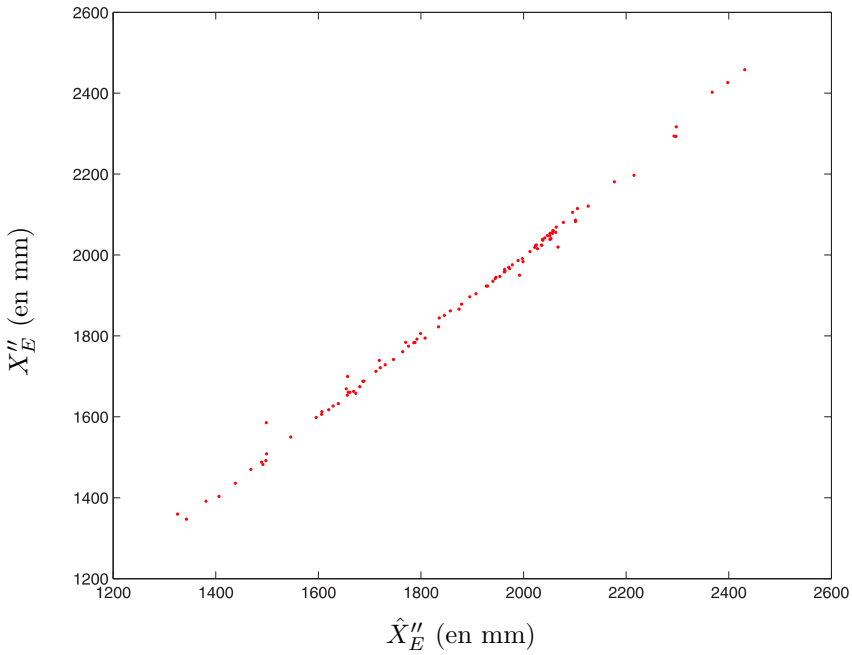


Fig. 4.16 *Relation* $X_E = f(\hat{X}_E)$

Cette courbe représente la sortie désirée en fonction de la sortie calculée. Les valeurs sont exprimées en millimètre.

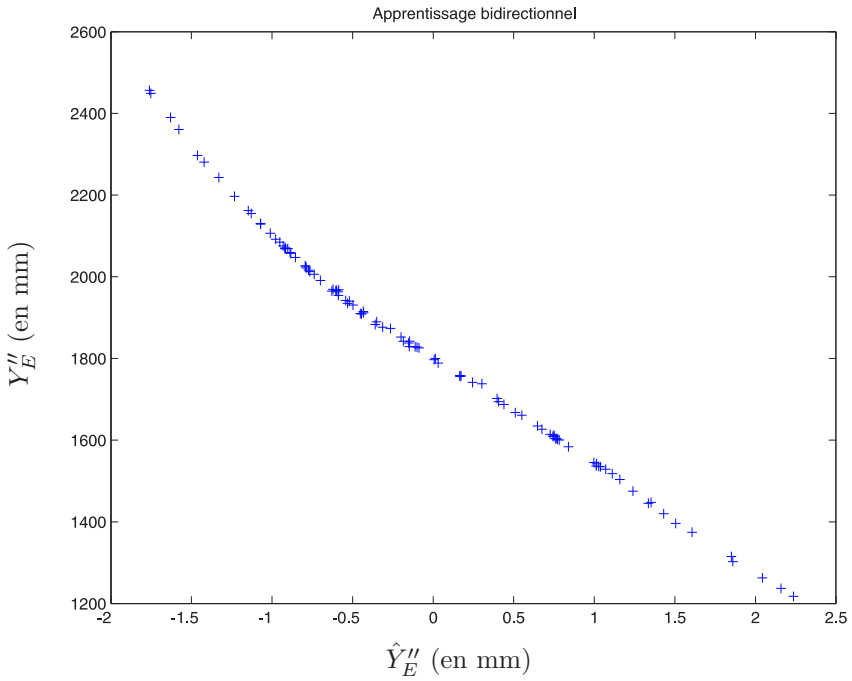


Fig. 4.17 *Relation* $Y''_E = f(\hat{Y}''_E)$

Cette courbe représente la sortie désirée en fonction de la sortie estimée. Deux caractéristiques peuvent être soulignées : le changement d'échelle et une légère courbure.

ordre de grandeur (pour que chacune participe de la même manière à la sélection du vainqueur).

L'étape suivante dans l'apprentissage des coordonnées projetées de l'effecteur sur $\mathcal{H}$, est l'apprentissage du bloc H.

Apprentissage du second bloc : détermination de h

Le second bloc de l'architecture modulaire permet de faire une estimation de la grandeur h, qui est, rappelons-le, la hauteur qui sépare le plan $\mathcal{F}$ du plan parallèle à celui-ci et passant par le point représentant l'effecteur. Cette grandeur est le lien qu'il y a entre Y'_E et Y_E. Le réseau chargé de l'apprentissage prend trois entrées, dont deux sont estimées par le bloc précédent, et estime une seule sortie. Les paramètres de ce réseau sont les mêmes que ceux du réseau précédent. Ils sont récapitulés dans la table 4.2.

La sortie désirée h nécessaire à l'apprentissage supervisé est une variable interne à la décomposition et n'est par conséquent pas mesurable. Nous introduisons un second bloc inverse, Inv H, dans notre architecture modulaire afin de faire une seconde estimation de la grandeur h (voir figure 4.15). Pour des raisons semblables à celles citées plus haut, le module inverse Inv H et le module H ont les mêmes caractéristiques d'apprentissage.

Le module inverse a trois entrées : HE la hauteur de l'effecteur dans l'espace 3D, Y_E la coordonnée de l'effecteur projetée sur l'axe y et α_t l'angle de tilt de la tête supportant les deux caméras. Ce sont les mêmes entrées que le bloc inverse précédent. Ces trois entrées permettent de déterminer h de la manière suivante :

$$h = \frac{HE - Y_E \tan(\alpha_t)}{cos(\alpha_t)} \quad . \tag{4.24}$$

Il est tout à fait envisageable de n'utiliser qu'un seul réseau pour apprendre les deux flux inverses. Il faut alors qu'il y ait deux sorties, l'une pour Y''_E et l'autre pour l'estimation de h. Nous avons fait le choix d'utiliser deux réseaux distincts afin de pouvoir les mêmes paramètres d'apprentissage et les mêmes évolutions de coefficients pour la paire P''_E/Inv P et pour la paire H/Inv H. Cela permet notamment de retarder l'apprentissage de la seconde paire, pour que ses entrées, issues d'un bloc précédent, soient représentatives des grandeurs souhaitées.

Le nuage de points que nous obtenons en traçant la relation $h = f(\hat{h})$ (voir figure 4.18), est relativement éloigné de ce que nous pouvions attendre. Deux possibilités s'offrent à nous pour l'interprétation : soit l'apprentissage a échoué, soit la relation apprise n'est pas exactement celle souhaitée.

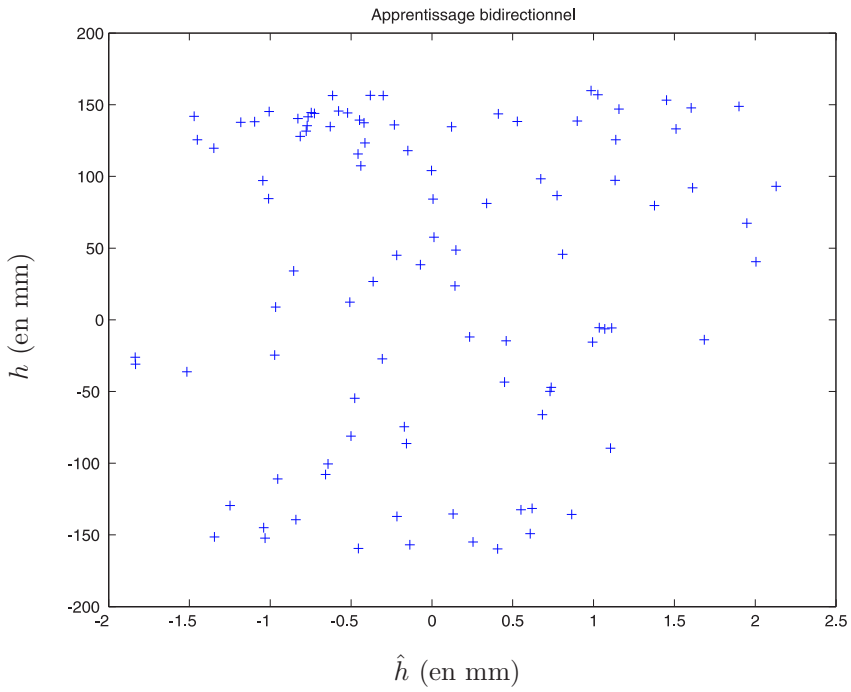


Fig. 4.18 *Sortie désirée en fonction de la sortie calculée*

Ce nuage de points représente la sortie désirée h en fonction de la sortie calculée. La sortie calculée est en fait une combinaison linéaire de h et de Y''_E.

Il faut bien garder à l'esprit que nous avons deux blocs qui doivent faire l'estimation d'une même grandeur, h. Le bloc H apprend une fonction f_1 :

$$\hat{h} = f_1(X''_E, Y''_E, I^e_{yg}) \ , \tag{4.25}$$

et le bloc inverse apprend une fonction f_2 :

$$\hat{h} = f_2(Y_E, HE, \alpha_t) \ . \tag{4.26}$$

Les seules contraintes imposées sont que la moyenne et la variance des réponses de chacune des fonctions tendent respectivement vers zéro et un. La grandeur estimée doit être solution de l'équation fonctionnelle $f_1() = f_2()$ quelque soient les variables de chacune des fonctions. Les connaissances que nous avons du problème nous laissent penser que la grandeur estimée par nos deux modules est une combinaison de h et de Y''_E :

Les réseaux de neurones permettent d'apprendre des relations linéaire ou non. Un réseau de neurones peut apprendre la relation de type $\hat{h} = f(h, Y''_E)$, si elle existe. La figure 4.19 montre le schéma d'apprentissage de ce module de test.

L'erreur d'estimation est inférieure au pourcent. La grandeur $\hat{h}$ estimée par les fonctions f_1 et f_2 est bien fonction de h et Y''_E.

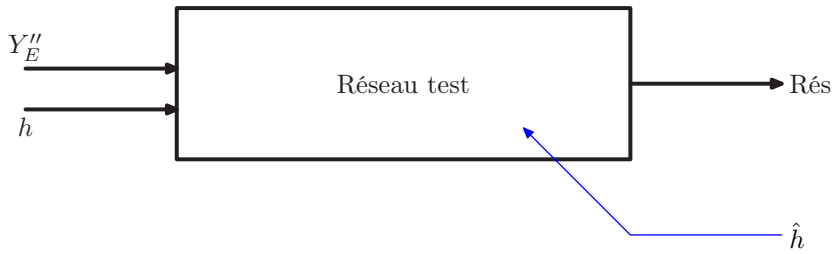


Fig. 4.19 *Réseau pour apprendre la relation entre h et Y''_E*

Nous allons utiliser les sorties calculées comme entrées du module suivant, qui constitue la dernière étape dans l'estimation de la coordonnée de l'effecteur sur l'axe y du plan horizontal.

Apprentissage du troisième bloc : projection sur $\mathcal{H}$

Le dernier bloc à apprendre de notre architecture modulaire est celui qui fournit les véritables coordonnées de l'effecteur sur le plan horizontal $\mathcal{H}$.

La coordonnée selon l'axe x est connue depuis le premier bloc de l'architecture. Rappelons que le passage du plan $\mathcal{F}$ au plan $\mathcal{H}$ n'affecte pas la coordonnée en x, il résulte simplement d'une rotation selon l'axe x.

Les trois entrées de ce module sont $\hat{Y}''_E$ la projection de la coordonnée Y de l'effecteur sur le plan $\mathcal{F}$ (issue du premier bloc), α_t l'angle formé entre les plans $\mathcal{F}$ et $\mathcal{H}$, $\hat{h}$ la grandeur intervenant dans le passage de Y'_E à Y_E (issue du second bloc). Comme nous l'avons vu plus haut, $\hat{h}$ n'est pas simplement une estimation de h, mais une relation plus complexe dans laquelle intervient h. La sortie de ce module est Y_E. Les paramètres d'apprentissage sont les mêmes que pour les modules précédents. Ces paramètres sont récapitulés dans la table 4.2.

Les résultats de l'apprentissage de ce bloc, et donc de toute l'architecture, sont montrés sur la figure 4.20. La courbe représentée est la sortie désirée en fonction de la sortie calculée : $Y_E = f(\hat{Y}_E)$.

Les résultats d'apprentissage de ce dernier bloc sont le fruit d'un apprentissage classique. La sortie désirée est connue et mesurable. Cela explique que la courbe obtenue peut être assimilée à une droite passant pas l'origine, ayant un coefficient directeur de un.

L'erreur moyenne d'estimation de Y_E pour le schéma complet est inférieure au pourcent, et est de l'ordre de 4% d'erreur maximale. Nous avons pu montrer que l'apprentissage d'une fonction complexe peut être effectué avec des cartes auto-organisatrices, si la tâche complexe est décomposée en sous-tâches.

Toutes les variables intervenant dans le calcul de θ_4 sont maintenant connues.

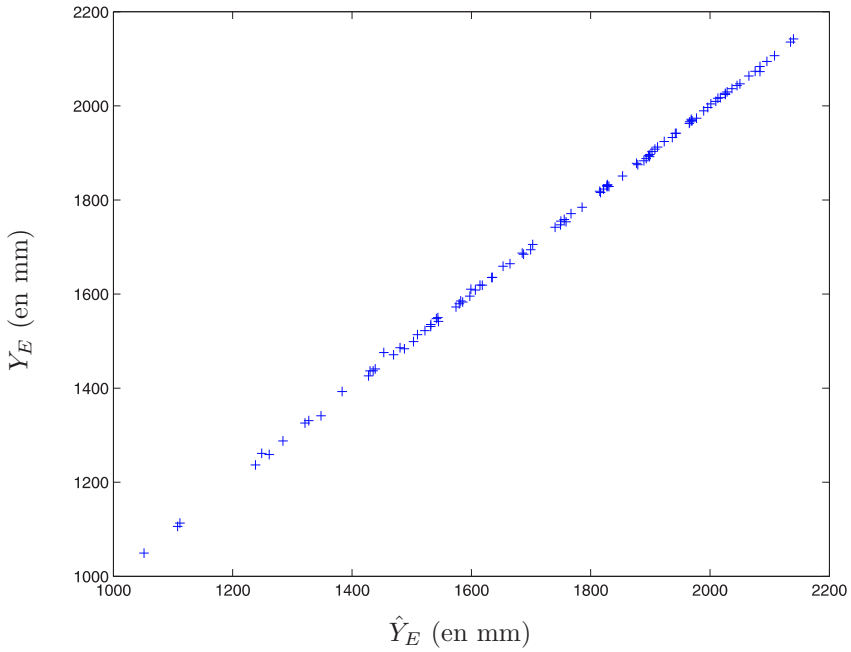


Fig. 4.20 *Relation* $Y_E = f(\hat{Y}_E)$

Cette courbe représente la sortie désirée $\hat{Y}_E$ *en fonction de la sortie calculée. Les valeurs sont exprimées en millimètre.*

γ.– Évaluation de θ_4

Les informations de la tête et visuelles ont permis de déterminer les grandeurs X_P, Y_P, X_E et Y_E nécessaires pour le calcul de θ_4. Les quatre grandeurs estimées permettent de déterminer l'_4, la longueur de la main projetée sur le plan horizontal $\mathcal{H}$. Il suffit maintenant d'appliquer la formule 4.4 pour déterminer les valeurs de θ_4, l'angle formé par la main par rapport au plan $\mathcal{H}$.

Les informations visuelles permettent de faire une estimation de l'orientation de la main avec une erreur moyenne de 1.5 degrés.

L'approche présentée ici permet d'estimer l'orientation de n'importe quel objet en utilisant les informations issues de la tête robotique (les angles qui permettent d'orienter les deux caméras et les informations images des deux extrémités de l'objet observé). Elle peut permettre, par exemple, d'estimer l'inclinaison d'un objet pour pouvoir ensuite déterminer une commande qui va permettre à un bras manipulateur d'appréhender correctement cet objet. Cette approche nécessite néanmoins de connaître la taille de l'objet observé mais s'affranchit de sa nature.

4.6 Discussions

L'objectif de ce chapitre est de montrer que les architectures modulaires apportent une solution à l'apprentissage de fonction complexe. Pour appuyer ce propos, nous faisons référence à une tâche robotique complexe. Elle consiste en l'estimation de l'orientation de l'effecteur d'un robot quatre axes. Pouvoir apprendre une tâche telle que celle-ci nécessite de subdiviser le problème en sous-tâches.

Bien souvent, la décomposition associée à une tâche n'est pas unique. Le contexte, par exemple, peut guider ce choix. La décomposition que nous avons choisie repose sur la géométrie du système.

Nous avons abordé le problème de la manière suivante. En fonction de l'angle à estimer, la projection de la main sur le plan horizontal varie. L'idée est de faire une estimation de cette longueur projetée, en utilisant les informations visuelles. Quatre coordonnées doivent être estimées. Parmi elles, trois résultent d'un apprentissage classique (en raison de la manière dont a été posé le problème). La dernière nécessite quant à elle l'emploi d'une architecture modulaire (en raison de la taille du vecteur d'entrées intervenant dans son estimation). Il est vrai que cette méthode ne renseigne pas sur le signe de l'orientation. Celui-ci peut aussi être déterminé en fonction des informations images.

Quelques remarques à propos de l'apprentissage doivent encore être émise. Nous avons, dans ce chapitre, développé les modèles de chaque module. Bien évidemment, cela n'a de sens que dans le cadre de notre étude. Ils servent simplement à valider notre approche. Il reste néanmoins indispensable de faire une étude préalable du problème, afin d'en dégager une décomposition, qui permet de fragmenter le vecteur d'entrées et d'insérer des connaissances a priori dans la structure.

Pour que tous les modules d'une architecture modulaire puissent être entraînés, il faut mettre en place un ou plusieurs flux inverses, de manière à avoir des estimations des grandeurs intermédiaires. Il est fort intéressant de remarquer que nous n'avons qu'une maîtrise toute relative de ces estimations. En effet, deux blocs, avec des entrées différentes, apprennent chacun une fonction qui doit fournir l'estimation d'une même grandeur. Nous en avons eu un exemple lors de l'estimation de h. La grandeur estimée était en fait une combinaison linéaire de h avec une autre variable, qui restait sommes toutes représentative de h. La convergence des deux modules est largement influencée par les contraintes utilisées (contraintes qui, rappelons-le, permettent de faire converger les deux algorithmes vers une valeur représentative, en la normalisant). Il sera intéressant d'étudier plus en avant l'influence des contraintes, sur la rapidité de la convergence ou encore sur la classe de solutions possibles. Il semble acquis que plus il y a de contraintes, moins y a de solutions possibles.

4.7 Conclusion

L'estimation de l'orientation de l'effecteur d'un robot quatre axes, en utilisant les informations images ainsi que la position et l'orientation des caméras, reste une tâche difficile. La multiplication des informations à traiter, comme par exemple l'observation de plusieurs éléments dans les images, augmente d'autant la dimentionnalité du problème. Ceci a pour conséquence directe, la nécessité d'utiliser plus de modules.

Nous avons proposé, dans ce chapitre, une architecture modulaire capable de résoudre cette tâche. La modularité s'avère être indispensable pour résolver un problème de dimensionnalité élevée comme celui-ci.

Pour valider l'approche neuronale modulaire, nous avons défini, afin de résoudre la tâche robotique, une décomposition s'appuyant sur la géométrie du système Hermann *et al.* (2004c,b).

Afin d'avoir des éléments de comparaison, nous avons développé les différents modèles nécessaires à la détermination des coordonnées projetées des deux extrémités de la main, c'est-à-dire de l'effecteur et du poignet. Nous avons fait ce travail en respectant la décomposition proposée.

Cette décomposition a donné naissance à un schéma d'apprentissage. Les différents blocs de l'architecture peuvent être entraînés les uns après les autres ou bien simultanément. Cette dernière technique, plus délicate à mettre en œuvre, est bien plus intéressante pour un apprentissage en ligne. Il faut alors porter un soin particulier au choix des variables d'apprentissage ainsi qu'à leur évolution.

Les résultats relatifs à chacun des modules de l'architecture ont été présentés et ont donné lieu à une discussion. Notamment sur l'estimation des variables intermédiaires, largement conditionnées par les contraintes appliquées pour faire converger l'apprentissage. Bien que la modularité nécessite une étude préalable pour pouvoir en dégager une décomposition, elle n'en reste pas moins une solution efficace et puissante pour rendre possible l'apprentissage de fonctions complexes avec des réseaux de neurones.

Conclusion

Contributions de la thèse

Dans cette thèse, nous avons utilisé des cartes auto-organisatrices pour apprendre des fonctions robotiques complexes. Ces réseaux sont régis par des règles simples qui permettent de les utiliser en ligne. De plus, les relations de voisinage dont ils disposent leur permet d'apprendre rapidement. Cependant, les cartes auto-organisatrices, quand elles sont utilisées pour l'apprentissage de fonctions complexes, souffrent de la dimensionnalité de l'espace d'entrées.

Nous avons pu voir que pour pouvoir être apprise, une tâche robotique complexe doit être décomposée. Les différentes formes de décomposition qui existent ont été exposées. Les architectures séquentielles répondent à nos attentes, à savoir réduire la dimensionnalité des réseaux utilisés.

Afin d'être résolue, une tâche doit faire l'objet d'une étude préliminaire qui permet de dégager une structure modulaire et de séparer les entrées. Une telle structure complique quelque peu l'apprentissage, du fait que des variables intermédiaires sont créées et doivent, par conséquent, être apprises. La méthode retenue et présentée pour réaliser l'apprentissage consiste à introduire des modules dans un flux inverse. Ils réalisent des estimations des grandeurs intermédiaires.

Afin d'illustrer les avantages à utiliser une structure bidirectionnelle pour l'apprentissage de fonctions complexes, nous avons présenté deux applications d'asservissement visuel adaptatif. La première présente une manière de définir les positions angulaires d'un bras robotique à trois axes et utilisant les informations visuelles issues d'une tête robotique surmontée de deux caméras. Les orientations des caméras sont quelconques. La seconde application propose une décomposition pour

déterminer l'orientation de l'effecteur d'un robot quatre axes. Ces deux exemples d'apprentissage modulaire peuvent servir à résoudre un problème plus général qui est la réalisation de la pose (c'est-à-dire la position et l'orientation) de l'effecteur d'un robot quatre axes, par un retour visuel.

Nous avons montré qu'il est possible de réaliser des tâches d'asservissement visuel adaptatif avec plusieurs cartes auto-organisatrices dans une architecture modulaire. Il faut pour cela que le problème soit décomposé et mis en œuvre dans une structure bidirectionnelle. Les approches modulaires peuvent trouver des applications dans d'autres domaines que la robotique.

Perspectives

Les perspectives envisagées pour ce travail sont multiples :

- Compléter le schéma de décomposition pour travailler sur un système robotique encore plus complexe, qui pourrait permettre de réaliser une pose complète. C'est-à-dire permettre d'orienter l'effecteur dans toutes les directions de l'espace.
- Étendre l'approche à d'autres type de réseaux de neurones.
- Étudier et analyser les contraintes appliquées aux modules qui font l'estimation d'une même grandeur. En effet, les contraintes utilisées dans nos applications concernent la moyenne et la variance. Il peut être intéressant d'étudier le rôle de ces contraintes sur la rapidité de convergence des algorithmes ou encore sur la classe de solutions possibles.

Bibliographie

C. Adda, N. Lefort-Piat et G. Laurent.
Micropositioning by pushing with control by reinforcement learning.
In *35th International Symposium on Robotics*, Paris-Nord Villepinte, France, 2004.
30

A. Alhaj, C. Collewet et F. Chaumette.
Visual servoing based on dynamic vision.
In *IEEE International Conference on Robotics and Automation (ICRA'03)*, volume 3, pages 3055–3060. Taipeh, Taiwan, 2003.
14

Y. Aloimonos, editor.
Active Perception.
Lawrence Erlbaum Associates, 1993.
13

M. Aupetit.
Robust topology representing networks.
In M. Verleysen, editor, *11th European Symposium on Artificial Neural Networks (ESANN'2003)*, pages 45–50, Bruges, Belgium, 2003. D-Facto.
39

L. Berthouze, P. Bakker et Y. Kuniyoshi.
Learning of oculo-motor control: A prelude to robotic imitation.
In *IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS'96)*, pages 376–381. Osaka, Japan, 1996.
22

L. Bottou et P. Gallinari.
A framework for the cooperation of learning algorithms.
In D. Touretzky et R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 781–788. Morgan Kaufmann, Denver, 1991.
80

S. Breton.
Une approche neuronale du contrôle robotique utilisant la vision binoculaire par reconstruction tridimensionelle.
Thèse de Doctorat, Université de Haute Alsace, 1999.
23, 33

J.-L. Buessler.
Architectures neuro-mimétiques modulaires, Application à l'asservissement visuel de systèmes robotiques.
Thèse de Doctorat, Université de Haute-Alsace, 1999.
3, 33, 44, 45, 51, 53, 61, 80

J. L. Buessler, R. Kara, P. Wira, H. Kihl et J. P. Urban.
Multiple self-organizing maps to facilitate the learning of visuo-motor correlations.
In *IEEE International Conference on Systems Man and Cybernetics*, volume 3, pages 470–475. IEEE Press, Tokyo, Japan, 1999.
23

J.-L. Buessler et J.-P. Urban.
Visually guided movements: learning with modular neural maps in robotics.
Neural Networks, Special Issue "Neural Control and Robotics : Biology and Technology", vol. 11(7–8) p. 1395–1415, 1998.
23, 55

J.-L. Buessler, J.-P. Urban et J. Gresser.
Additive composition of supervised self-organized maps.
Neural Processing Letters, vol. 15(1) p. 9–20, 2002.
23, 55

J.-L. Buessler, J.-P. Urban, G. Hermann et H. Kihl.
Color histogram similarity for robot-arm guiding.
In *International Conference on Complex Systems, Intelligence and Modern Technology Applications (CSIMTA 2004)*, pages 515–519. Cherbourg, France, 2004.
11, 32

T. Caelli, L. Guan et W. Wen.
Modularity in neural computing.
Proceedings of the IEEE, Special Issue on Computational Intelligence, vol. 87(9) p. 1497–1518, 1999.
45

G. J. Chappell et J. G. Taylor.
The temporal kohonen map.
Neural Networks, vol. 6(3) p. 441–445, 1993.
44

F. Chaumette.
La relation vision-commande: théorie et application à des tâches robotiques.
Thèse de Doctorat, Université de Rennes 1, IRISA, 1990.
12, 20

F. Chaumette.
De la perception à l'action : l'asservissement visuel, de l'action à la perception : la vision active.
Habilitation à diriger des recherches, Université de Rennes 1, Institut de Formation Supérieure en Informatique et en communication, 1998.
13, 16

F. Chaumette.
Avancées récentes en asservissement visuel.
In *Journées nationales de la recherche en robotique (JNRR'03)*, pages 103–108. Clermont-Ferrand, 2003.
9

S.-B. Cho.
Evolving multiple sensory-motor controllers based on cellular neural network.
In *International Fuzzy System Association World Congress (IFSA2001)*, Vancouver, Canada, 2001.
55

D.-I. Choi et S.-H. Park.
Self-creating and organizing neural networks.
IEEE Transactions on Neural Networks, vol. 5 p. 561–575, 1994.
44

P. Coiffet.
La Robotique : principe et applications.
Hermès, Paris, 1992.
18, 77

A. I. Comport, E. Marchand et F. Chaumette.
A real-time tracker for markerless augmented reality.
In *Int. Symp. on Mixed and Augmented Reality, ISMAR'03*, pages 36–45, Tokyo, Japon, 2003.
10

P. I. Corke.
Visual control of robot manipulators - a review.
In K. Hashimoto, editor, *Visual Servoing*, volume 7 of *Robotics and Automated Systems*, pages 1–31. World Scientific, Singapore, 1993.
12

C. Corne.
Parallélisation de réseaux de neurones sur architecture distribuée.
Thèse de Doctorat, Université de Haute-Alsace, 1999.
45

K. Doya, K. Samejima, K. i. Katagiri et M. Kawato.
Multiple model-based reinforcement learning.
Neural Computation, vol. 14(6) p. 1347–1369, 2002.
56

G. Dreyfus, J. M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria et L. Hérault.
Réseaux de neurones : méthodologie et applications.
Eyrolles, Paris, 2002.
27

E. Erwin, K. Obermayer et K. Schulten.
Self-organizing maps: Ordering, convergence properties and energy functions.
Biological Cybernetics, vol. 67(1) p. 47–55, 1992.
40

N. R. Euliano et J. C. Principe.
Spatio-temporal self-organizing feature maps.
In *International Conference on Neural Networks*, volume 4, pages 1900–1905. 1996.
44

J. Feddema et O. Mitchell.
Vision-guided servoing with feature-based trajectory generation.
IEEE Transactions on Robotics and Automation, vol. 5 p. 691–700, 1989.
12

J. Gangloff.
Asservissements visuels rapides d'un robot manipulateur à six degrés de liberté - Suivi de cible et de profilé.
Thèse de Doctorat, Université Louis Pasteur, 1999.
17

J. A. Gangloff et M. F. Mathelin.
Visual servoing of a 6-dof manipulator for unknown 3-d profile following.
IEEE Transactions on Robotics and Automation, vol. 18(4) p. 511–520, 2002.
17

N. R. Gans, P. I. Corke et S. A. Hutchinson.
Performance tests of partitioned approaches to visual servo control.
In *IEEE International Conference on Robotics and Automation (ICRA'02)*, pages 1616–1623, Washington, D.C., 2002.
17

H. Gomi et M. Kawato.
Recognition of manipulated objetcs by motor learning with modular architecture networks.
Neural Networks, vol. 6(4) p. 485–497, 1993.
55

T. Hamel et R. Mahony.
Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach.
IEEE Transactions on Robotics and Automation, vol. 18(2) p. 187–198, 2002.
15

M. Haruno, D. M. Wolpert et M. Kawato.
Mosaic model for sensorimotor learning and control.
Neural Computation, vol. 13(10) p. 2201–2220, 2001.
55

K. Hashimoto, editor.
Visual Servoing, volume 7.
World Scientific, Singapore, 1993.
12

S. Haykin.
Neural networks : A comprehensive Foundation, 2nd edition.
Prentice Hall, Upper Saddle River, N.J., 1999.
27, 30

G. Hermann.
Asservissement visuel d'une tête robotique avec un filtre de kalman.
Mémoire de dea, Université de Haute Alsace, Laboratoire MIPS, 2000.
10

G. Hermann, D. Gréboval, H. Kihl et J.-P. Urban.
Evaluation of color-based techniques for robotic positioning tasks.
In *8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004)*. Orlando, Florida, USA, 2004a.
11, 32

G. Hermann, P. Wira et J.-L. Buessler.
Organisation de réseaux de neurones pour l'apprentissage de fonctions robotiques complexes.
In *Journée du Centre de Recherche et d'Enseignements en Sciences Pour l'Ingénieur (CRESPIM'2003)*, pages 97–102. Mulhouse, 2003a.
79, 87

G. Hermann, P. Wira, J.-L. Buessler et J.-P. Urban.
Une approche neuronale modulaire pour l'estimation de l'orientation de l'effecteur d'un robot 4 axes.
In *Dix-huitième Journées des Jeunes Chercheurs en Robotique (JJCR'18)*. Douai, 2004b.
113

G. Hermann, P. Wira et J.-P. Urban.
Neural networks organizations to learn complex robotic functions.
In M. Verleysen, editor, *11th European Symposium on Artificial Neural Networks (ESANN'2003)*, pages 33–38. D-Facto, Bruges, Belgium, 2003b.
23, 84, 87

G. Hermann, P. Wira et J.-P. Urban.
A modular learning architecture for orienting a robot in a visual servoing task.
In *NeuroBotics Workshop (KI'2004)*. Ulm, Germany, 2004c.
23, 113

M. Herrmann, H.-U. Bauer et T. Villmann.
Topology preservation in neural maps.
In *European Symposium on Artificial Neural Networks (ESANN'97)*, pages 205–210. 1997.
39

L. Hetel, J.-L. Buessler et J.-P. Urban.
Superposition-based order analysis with self-organizing maps.
In *International Joint Conference on Neural Networks (IJCNN 2004)*. Budapest, Hungary, 2004.
39

K. Hosoda et M. Asada.
Versatile visual servoing without knowledge of true jacobian.
In *International Conference on Intelligent Robots and Systems (IROS'94)*, pages 186–193, Müchen, Germany, 1994.
22

S. Hutchinson, G. Hager et P. Corke.
A tutorial on visual servo control.
IEEE Transactions on Robotics and Automation, vol. 12(5) p. 651–670, 1996.
12, 20

IEEE RA.
Ieee transactions on robotics and automation.
Special Issue on Vision-Based Control of Robot Manipulators, vol. 12(5), 1996.
12

IEEE RA.
Ieee transactions on robotics and automation.
Special Issue on Medical Robotics, vol. 19(5), 2003.
12

IEEE SMC.
Ieee transactions on systems, man, and cybernetics-part b.
Special Issue on Learning Autonomous Robots, vol. 26(3), 1996.
31

R. A. Jacobs et M. I. Jordan.
Learning piecewise control strategies in a modular neural network architecture.
IEEE Transactions on Systems, Man, and Cybernetics, vol. 23(2) p. 337–345, 1993.
47, 55

R. A. Jacobs, M. I. Jordan, S. J. Nowlan et G. E. Hinton.
Adaptive mixtures of local experts.
Neural Computation, vol. 3(1) p. 79–87, 1991.
46

W. Jacquet.
Paradigmes connexionnistes pour l'apprentissage de fonctions robotiques : application aux problèmes d'asservissement visuel de bras manipulateur.
Thèse de Doctorat, Université de Haute-Alsace, 2001.
33

M. Jägersand.
Visual servoing using trust region methods and estimation of the full coupled visual-motor jacobian.
In *International Conference on Applications of Robotics and Control*, pages 105–108. IASTED, 1996.
21, 22

Q. Ji et S. Dai.
Self-calibration of a rotating camera with a translational offset.
IEEE Transactions on Robotics and Automation, vol. 20(1) p. 1–14, 2004.
23

R. Kara.
Une approche modulaire du réseau de neurones CMAC pour la commande d'un système robot-vision.
Thèse de Doctorat, Université de Haute-Alsace, 2003.
33

M. Kawato.
Adaptation and learning in control of voluntary movement by the central nervous system.
Advanced Robotics, vol. 3(3) p. 229–249, 1989.
55

M. Kawato, M. Isobe, Y. Maeda et R. Suzuki.
Coordinates transformation and learning control for visually-guided voluntary movement with iteration: A newton-like method in a function space.
Biological Cybernetics, vol. 59 p. 161–177, 1988.
55

M. Kawato, Y. Maeda, Y. Uno et R. Suzuki.
Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion.
Biological Cybernetics, vol. 62 p. 275–288, 1990.
55

M. Kawato, Y. Uno, M. Isobe et R. Suzuki.
A hierarchical model for voluntary movement and its application to robotics.
In *IEEE International Conference on Neural Networks*, volume 4, pages 573–582.
IEEE Press, San Diego, CA, 1987.
31, 55

T. Kohonen.
Analysis of a simple self-organizing process.
Biological Cybernetics, vol. 44(2) p. 135–140, 1982a.
34

T. Kohonen.
Self-organized formation of topologically correct feature maps.
Biological Cybernetics, vol. 43(1) p. 59–69, 1982b.
34, 36

T. Kohonen.
Self-Organization and Associative Memory, 3rd edition, volume 8.
Springer-Verlag, Berlin, 1984.
34

T. Kohonen.
Self-Organizing Maps, volume 30.
Springer-Verlag, Berlin, 1995.
36, 44, 45, 49

T. Kohonen.
Emergence of invariant-feature detectors in the adaptative-subspace som.
Biological Cybernetics, vol. 75 p. 281–291, 1996.
45

T. Kohonen, S. Kaski et H. Lappalainen.
Self-organized formation of various invariant-feature filters in the adaptative-subspace som.
Neural Computation, vol. 9(6) p. 1321–1344, 1997.
45

D. Kragic.
Visual Servoing for Manipulation: Robustness and Integration Issues.
Thèse de Doctorat, Royal Institute of Technology, Computational Vision and Active Perception (CVAP), 2001.
21

D. Kragic et H. I. Christensen.
Robust visual servoing.
International Journal of Robotics Research - Special Issue on Visual Servoing, vol. 22(10-11) p. 923–940, 2003.
21

A. Krupa.
Commande par vision d'un robot de chirurgie laparoscopique.
Thèse de Doctorat, Institut National Polytechnique de Lorraine, 2003.
12

D. Kuhn.
Une approche neuronale pour l'asservissement visuel d'un robot manipulateur.
Thèse de Doctorat, Université de Haute-Alsace, 1997.
22, 32, 55

M. Kuperstein.
Neural model for adaptive hand-eye coordination for single postures.
Sciences, vol. 239 p. 1308–1311, 1988.
31

J.-P. Lallemand et S. Zeghloul.
Robotique : aspects fondamentaux, modélisation mécanique, CAO robotique, commande avec exercices corrigés.
Masson, Paris, 1994.
18, 77

Y. F. Li et R. S. Lu.
Uncalibrated euclidean 3-d reconstruction using an active vision system.
IEEE Transactions on Robotics and Automation, vol. 20(1) p. 15–25, 2004.
23

E. López-Rubio, J. e. Muñoz Pérez et J. e. A. Gómez-Ruiz.
A principal components analysis self-organizing map.
Neural Networks, vol. 17(1) p. 261–270, 2004.
45

E. Malis.
Contribution à la modélisation et à la commande en asservissement visuel.
Thèse de Doctorat, Université de Rennes I, 1998.
15

E. Malis.
Visual servoing invariant to changes in camera-intrinsic parameters.
IEEE Transactions on Robotics and Automation, vol. 20(1) p. 72–81, 2004.
13

E. Malis et F. Chaumette.
Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods.
IEEE Transactions on Robotics and Automation, vol. 18(2) p. 176–186, 2002.
16

P. Martinet.
Asservissement visuel.
Habilitation à diriger des recherches, Université Blaise Pascal, LASMEA, 1999.
12

T. M. Martinetz et K. J. Schulten.
A 'neural-gas' network learns topologies.
In T. Kohonen, K. Mäkisara, O. Simula et J. Kangas, editors, *International Conference on Artificial Neural Networks*, volume 1, pages 397–402. Elsevier, Espoo, Finland, 1991.
34

T. M. Martinetz et K. J. Schulten.
Topology representing networks.
Neural Networks, vol. 7(3) p. 507–522, 1994.
39

Y. Mezouar et F. Chaumette.
Path planning for robust image-based control.
IEEE Transactions on Robotics and Automation, vol. 18(4) p. 534–549, 2002.
16, 21

Y. Mezouar et F. Chaumette.
Optimal camera trajectory with image-based control.
International Journal of Robotics Research, vol. 22(10) p. 781–804, 2003.
16, 21

R. C. Miall.
Modular motor learning.
Trend in Cognitive Sciences, vol. 6(1) p. 1–3, 2002.
56

W. T. Miller III.
Real-time application of neural networks for sensor-based control of robots with vision.
IEEE Transactions on Systems, Man, and Cybernetics, vol. 19(4) p. 825–831, 1989.
22

W. T. Miller III, R. S. Sutton et P. J. Werbos, editors.
Neural Networks for Control.
The MIT Press, Cambridge, MA, 1990.
22, 31

H. Miyamoto et M. Kawato.
A tennis serve and upswing learning robot based on bi-directional theory.
Neural Networks, vol. 11(7-8) p. 1331–1344, 1998.
23

H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada et M. Kawato.
A kendama learning robot based on bi-directional theory.
Neural Networks, vol. 9(8) p. 1281–1302, 1996.
23, 55

V. Murino.
Structured neural networks for pattern recognition.
IEEE Transactions on Systems, Man, and Cybernetics, vol. 28(4) p. 553–561, 1998.
50

NN.
Neural networks.
Special Issue on Neural Control and Robotics : Biology and Technology, vol. 11 (7-8), 1998.
31

NN.
Neural networks.
Special Issue on New Developments on Self-Organizing Maps, vol. 15(8-9), 2002.
44

R. P. Paul.
Robot Manipulators: mathematics, Programming, and Control.
MIT Press, 1981.
61

J. A. Piepmeier, G. V. McMurray et H. Lipkin.
Uncalibrated dynamic visual servoing.
IEEE Transactions on Robotics and Automation, vol. 20(1) p. 143–147, 2004.
23

J. C. Principe, L. Wang et M. Motter.
Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control.
Proceedings of the IEEE, Special Issue on : Intelligent Signal Processing, edited by S. Haykin, B. Kosko, vol. 86(11) p. 2240–2258, 1998.
42

RAS.
Robotics and autonomous systems.
Special Issue on Perceptual Anchoring, vol. 43(2–3), 2003.
31

H. J. Ritter, T. M. Martinetz et K. J. Schulten.
Neural Computation and Self-Organizing Maps.
Addison-Wesley, Reading, MA, 1992.
21, 22, 30, 36, 49

H. J. Ritter et K. J. Schulten.
Topology-conserving mappings for learning motor tasks.
In *International Conference on Neural Networks for Computing*, pages 376–380, Snowbird, Utah, 1986.
22, 36, 41

H. J. Ritter et K. J. Schulten.
Extending kohonen's self-organizing mapping algorithm to learn ballistic movements.
In R. Eckmiller et C. von der Malsburg, editors, *Neural Computers*, pages 393–406. Springer-Verlag, Berlin, 1988.
22, 42

E. Ronco et P. J. Gawthrop.
Modular neural networks: State of the art.
Technical report csc-95026, University of Glasgow, 1995.
45

K. Samejima, K. Doya et M. Kawato.
Inter-module credit assignment in modular reinforcement learning.
Neural Networks, vol. 16(7) p. 985–994, 2003.
56

A. C. Sanderson et L. E. Weiss.
Image based visual servo control using relationnal graph error signal.
In M. A. I. SMC, editor, *Internationnal Conference on Cybernetics Graph Error Signal*, pages 1074–1077, Cambridge, 1980.
12

A. J. C. Sharkey.
On combining artificial neural nets.
Connection Science, vol. 8 p. 299–314, 1996.
45

A. J. C. Sharkey.
Modularity, combining and artificial neural nets.
Connection Science, vol. 9 p. 3–10, 1997.
45

S. M. Stringer, E. T. Rolls et T. P. Trappenberg.
Self-organising continuous attractor networks with multiple activity packets, and the representation of space.
Neural Networks, vol. 17(1) p. 5–27, 2004.
44

S. M. Stringer, E. T. Rolls, T. P. Trappenberg et I. E. T. de Araujo.
Self-organizing continuous attractor networks and motor function.
Neural Networks, vol. 16(2) p. 161–182, 2003.
44

M. Swain et D. Ballard.
Color indexing.
International Journal of Computer Vision, vol. 7(1) p. 11–32, 1991.
10

J. P. Urban.
Une approche de type asservissement visuel appliquée à la robotique.
Thèse de Doctorat, Université Blaise Pascal, 1990.
12

J. P. Urban, J.-L. Buessler et J. Gresser.
Neural networks for visual servoing in robotics soft computing for intelligent robotic systems.
Series Studies in Fuzziness and Soft Computing, pages 81–111. Heidelberg, 1998.
21, 22

J. P. Urban, J. L. Buessler et P. Wira.
Neuromodule-based visual servoing of a robot arm with a 2 d.o.f. camera.
In *IEEE International Conference on Systems Man and Cybernetics*, volume 4, pages 3507–3512. IEEE Press, Orlando, 1997.
55

T. Villmann, R. Der et T. Martinetz.
Topology preservation in self-organizing feature maps: exact definition and measurement.
IEEE Transactions on Neural Networks, vol. 8(2) p. 256–266, 1997.
39

T. Voegtlin.
Recursive self-organizing maps.
Neural Networks, Special Issue "New Developments in Self-Organizing Maps", vol. 15(8-9) p. 979–991, 2002.
44

Y. Wada et M. Kawato.
A neural networks model for arm trajectory formation using forward and inverse dynamics models.
Neural Networks, vol. 6(7) p. 912–932, 1993.
48, 55

L. E. Weiss.
Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach.
Thèse de Doctorat, Carnegie-Mellon University, 1984.
CMU-RI-TR-84-16.
12, 20

B. Widrow et M. A. Lehr.
30 years of adaptative neural networks: Perceptron, madaline and backpropagation.
In *Proceedings of the IEEE*, volume 78, pages 1415–1442, 1990.
42

B. Widrow et E. Walach.
Adaptive Inverse Control.
Prentice Hall Press, Upper Saddle River, 1996.
42

P. Wira.
Réseaux neuromimétiques, modularité et statistiques : estimation du mouvement pour l'asservissement visuel de robots.
Thèse de Doctorat, Université de Haute-Alsace, 2002.
10, 33, 47, 55

D. M. Wolpert et M. Kawato.
Multiple paired forward and inverse models for motor control.
Neural Networks, Special Issue "Neural Control and Robotics : Biology and Technology", vol. 11(7-8) p. 1317–1329, 1998.
23, 46, 55

P. Zanne.
Contributions à l'asservissement visuel.
Thèse de Doctorat, Université Louis Pasteur, 2003.
9, 17

Modular neural-networks approach for a robot–vision system

Abstract:

The laboratory works on the neural network control of a robot–vision system. This thesis concerns the application of artificial neural networks to control a robotic arm with visual servoing. The study focuses on modular learning to carry out neural networks controllers.

We used self organizing maps to learn robotic functions which had been chosen for the simplicity of their algorithm as well as their capacity of generalization. These advantages enable us to have an online learning process, in real time. However these map can't deal properly with the important size of the input space following from the complexity of the task. To resolve the dimensionality problem we propose to divide the task into modules, each of them being composed of small-size artificial neural networks.

With a sequential decomposition the learning phase of the whole structure becomes more difficult. Indeed not all desired inputs and outputs (which are examples for the learning process) of each module are available. Therefore a learning process is enabled by introducing the modules into the inverse flow which can estimate other introduced intermediary variables. The learning process in then bidirectional.

The modular approach can therefore be considered as a neural network controller. The visual servoing has been validated on two complex robotic tasks: determining the angular configuration of a three-axis-robot and estimating the effector orientation of a four-axis-robot.

Keywords:

Bidirectionality, complex systems, decomposition, modularity, robotic system, self organizing maps, sequential architecture.

Approche neuromimétique modulaire pour la commande d'un système robot–vision

Résumé :

Les travaux du laboratoire se concentrent autour du contrôle neuromimétique d'une plate-forme robot-vision. Dans ce cadre, ce travail de thèse concerne l'application des réseaux de neurones artificiels à la commande d'un bras robotique par asservissement visuel. Cette étude porte plus particulièrement sur l'apprentissage modulaire, afin de réaliser des contrôleurs neuromimétiques.

Nous avons utilisé des cartes auto-organisatrices pour apprendre des fonctions robotiques. Celle-ci ont été choisies pour la simplicité de leur algorithme et leur capacité de généralisation, avantages qui nous permettent d'exécuter un apprentissage en ligne, en temps réel. Cependant, ces cartes souffrent de la dimensionnalité de l'espace d'entrées, liée à la complexité de la tâche à accomplir. Face à ce problème de dimensionnalité, nous proposons de décomposer la tâche en modules, chacun des modules étant alors constitué de réseaux de neurones artificiels de dimensionnalité plus petite.

La décomposition séquentielle rend la phase d'apprentissage de la structure complète plus délicate. En effet, toutes les entrées et toutes les sorties désirées (qui constituent les exemples d'apprentissage) de chacun des modules ne sont pas disponibles. L'apprentissage est alors rendu possible en introduisant d'autres modules dans le flux inverse, capables de faire d'autres estimations des variables intermédiaires introduites. L'apprentissage devient alors bidirectionnel.

L'approche modulaire proposée peut alors être considérée comme un contrôleur neuromimétique. L'asservissement visuel à été validé en simulation sur deux tâches robotiques complexes : la détermination de la configuration angulaire d'un robot trois axes et l'estimation de l'orientation de l'effecteur d'un robot quatre axes.

Mots-clés :

Architecture séquentielle, bidirectionnalité, cartes auto-organisatrices, décomposition, système complexe, système robotique, modularité.

Discipline :

E.E.A.– Génie informatique, automatique et traitement du signal

Laboratoire MIPS – Trop
Faculté des Sciences et Techniques, 4 rue des Frères Lumière, 68093 MULHOUSE Cedex