Apprentissage ouvert de representations et de fonctionnalites en robotique : anayse, modeles et implementation

Williams Paquier

Thèse

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du
Doctorat de l'Université Paul Sabatier de Toulouse
Spécialité : Robotique

par
Williams Paquier

Apprentissage Ouvert de Représentations et de Fonctionnalités en Robotique : Analyse, Modèles et Implémentation

---

Dario FLOREANO	EPFL, Lausanne (Rapporteur)
Agnès GUILLOT	LIP6, Paris (Rapporteur)
Raja CHATILA	LAAS, Toulouse (Directeur de thèse)
Philippe GAUSSIER	ETIS, Cergy-Pontoise
George GIRALT	LAAS, Toulouse
Michel COURDESSES	LAAS, Toulouse

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

À la mémoire de Laurence

Remerciements

Ce mémoire est la synthèse de réflexions menées depuis plusieurs années autour des questions des représentations et du traitement de l'information. Je tiens à remercier en premier lieu mon directeur de recherche, Raja Chatila, pour la confiance et le soutien qu'il m'a accordés tout au long de ma thèse. Bien que ces idées aient été concrétisées au LAAS-CNRS dans l'équipe Robotique et Intelligence artificielle, un long cheminement au sein de plusieurs laboratoires m'a permis de les développer et de les amener à terme.

Je souhaite vivement remercier monsieur Mohamed Ibnkahla, alors Postdoc au LEN7, pour m'avoir accueilli, encadré et surtout communiqué son goût et sa passion pour la recherche.

Je veux remercier Monsieur Michel Imbert, Monsieur Simon Thorpe et Monsieur Kevin O'Regan pour m'avoir transmis leur enthousiasme pour la recherche. Leurs cours de DEA, leurs séminaires et leurs conseils précieux ont fortement contribué à l'orientation de mon travail de thèse.

Durant mon séjour au CERCO, j'ai également eu la chance de rencontrer Lotfi Merabet, Denis Fize, et Rufin Van Rullen. Les longues discussions que nous avons eues ont peu à peu permis de clarifier mes idées sur les travaux que j'allais réaliser durant ma thèse. Qu'ils en soient remerciés.

Je souhaite vivement remercier Madame Agnès Guillot et Monsieur Dario Floreano pour m'avoir fait l'honneur d'être rapporteur cette thèse, Monsieur Michel Courdesses pour les conseils précieux qu'il m'a apportés et pour avoir accepté d'être membre de mon jury, Monsieur Philippe Gaussier pour avoir accepté d'être membre de mon jury, ainsi que Monsieur Georges Giralt pour avoir accepté de juger mon travail et de présider mon jury.

Je suis très reconnaissant à Nicolas Do Huu pour les échanges passionnés que nous avons eus au cours de ces deux dernières années. Ces discussions ont permis de mieux définir les concepts exposés dans cette thèse et ont également ouvert de nombreuses perspectives sur les travaux futurs.

Je tiens également à remercier Frédéric Py[1], Anthony Mallet et Aurélien Kamel pour leur importante contribution au développement de NeuSter.

Je souhaite remercier monsieur Hamid Aichoune pour m'avoir fait découvrir les logiciel libre et le monde open-source.

[1] voir Figure V.5

Je tiens à remercier mes camarades et amis du LAAS, Emmanuel Zenou, Solange Lemai, Rémi Driancourt, Benoit Morisset et Félix Ingrand dont certains ont également été des compagnons de rédaction : Romain Trinquart, Julien Pettré et Il-Kyun Jung.

Je tiens tout particulièrement à remercier Jackie Som et Marc Vaisset pour la relecture fastidieuse de ce manuscrit.

Enfin, je souhaite vivement remercier mes parents pour leurs soutiens et leur amour.

Table des matières

Table des figures

Liste des tableaux

Avant-propos

“Computers are useless...
They can only give you answers.”
Pablo Picasso

La majorité des systèmes robotiques autonomes développés aujourd’hui sont conçus dans le but d’exécuter un ensemble de tâches. Nous pouvons citer les robots autonomes pour l’exploration planétaire, les systèmes de navigation autonomes, ou encore les robots d’assistance. Ils ont une connaissance explicite du monde donnée par les concepteurs lors de la phase de développement et un ensemble de buts prédéterminés. Leur interaction avec le monde se résume à une observation et une série d’actions destinées à atteindre un but. Ils ne changent pas la configuration de leur environnement, ne construisent pas de nouveaux objets, ou, s’ils le font, ils ont au préalable toutes les informations sur la forme de l’objet qu’ils doivent obtenir.

Au contraire, la notion de système ouvert implique un développement qui ne soit pas guidé par un but préalablement et explicitement fixé. Nous proposons dans cette thèse l’analyse, la modélisation et l’implémentation d’un tel système. La contrainte d’ouverture pose directement la question de la représentation et de la fonctionnalité. La phase de développement n’est pas centrée sur la question de l’objectif final de la machine, mais comment la machine peut se donner elle-même une série de buts. Il n’existe pas aujourd’hui de théories couvrant à la fois l’acquisition autonome et ouverte de représentation et de fonctionnalité. Nous entendons par là un système de représentation suffisamment puissant pour rendre compte de la complexité du monde. C’est-à-dire un système qui puisse exprimer la richesse des catégories susceptibles d’être rencontrées et de les accompagner aussi loin que possible. Nous souhaitons étudier la capacité d’apprentissage de ces représentations. Le système autonome que nous allons étudier doit construire incrémentalement ses représentations du monde, sans intervention humaine. Rien ou très peu de choses doivent être initialement spécifiées. Une partie de l’objet de ce travail réside dans l’étude de l’acquisition autonome de connaissance. L’ensemble des fonctionnalités d’un tel système doit également croître incrémentalement. Les fonctionnalités d’un tel système peuvent être considérées comme des séquences d’actions dont l’objectif est d’activer un ensemble de représentations désirées. Ce système est autonome, il doit lui-même choisir les situations de l’environnement qu’il devra retrouver. Il faut donc comprendre comment un tel système peut lui-même choisir la façon dont ses comportements et ses compétences vont se

développer. Étudier et développer un système robotique de ce type impose un paradigme fondamentalement différent : nous ne programmons plus la machine pour qu'elle résolve une tâche, mais pour qu'elle apprenne à trouver des tâches à résoudre, sans nécessairement l'orienter vers celles-ci.

La principale contribution de ce travail est de proposer un système de codage ouvert de représentation. Ce système de représentation permet d'envisager des classes de systèmes capables d'apprendre leurs fonctionnalités de façon autonome.

Justifier une intuition a posteriori n'est pas trivial. Comment présenter le cheminement intellectuel qui a conduit à cette façon de considérer les systèmes ouverts ? Lorsque l'on souhaite explorer une nouvelle voie, on ne sait pas initialement sur quelles bases théoriques l'appuyer : aucune hypothèse ne doit donc être écartée. Nous proposons dans un premier temps un panorama des systèmes de représentations et de fonctionnalités en robotique. Nous montrons qu'il n'existe pas aujourd'hui de système qui puisse apprendre de façon autonome de nouvelles représentations. Quelles que soient les approches, un but ou un objectif est toujours fixé lors de la phase de conception ou d'utilisation. Cette façon de procéder est accompagnée par une série de modélisations spécifiques. Les systèmes ainsi développés sont difficiles à intégrer entre eux. L'analyse d'image, la navigation autonome, c'est-à-dire la capacité de construire une carte et de se localiser dans cette carte ou encore l'évitement d'obstacle, la planification, la manipulation, la reconnaissance ou la production de la parole, l'exécution de plan sont autant de théories qui apportent leur propre système de représentation (symbolique, géométrique, etc.) et rendent très complexe l'intégration de tous ces systèmes. Comme nous l'avons dit, l'apprentissage revient souvent à l'optimisation d'une fonctionnalité spécifiée en amont. Un système ouvert doit construire lui-même ses représentations et ses modèles de fonctionnement. Pour que la machine ait la possibilité de construire elle-même son système de présentation et de lier toutes les compétences qu'elle a acquises de façon autonome, une méthode homogène doit être envisagée.

Dans le chapitre II, nous analysons les conditions nécessaires d'un système de représentation et de fonctionnalité ouvert. Nous commençons par préciser les hypothèses de travail qui vont servir de fondement au système étudié, en particulier le fait que l'agent étudié soit hédoniste, incarné et situé. Les propriétés nécessaires et suffisantes à la réalisation d'un tel système sont dérivées d'une expérience extrêmement simple qui met en jeu un agent hédoniste capable de généraliser ses actions en utilisant un système de représentation distribué. Nous précisons bien que certaines propriétés ne sont pas justifiables par un raisonnement, mais semblent suffisantes. L'une des propriétés fondamentales issue de cette analyse est la nécessité pour un tel système d'abstraire la notion de sémantique. Une machine qui doit créer de la connaissance de façon autonome ne peut construire elle-même ses représentations sous forme symbolique. À plus forte raison si elle part d'un ensemble de représentations quasi nul. Ce type d'approche vise à apporter des réponses au problème de l'ancrage soulevé par Harnard [Harnad 90]. Partir des données brutes permet de créer un système de représentation mieux adapté à l'environnement.

Un problème similaire à déjà été résolu dans l'histoire des sciences. L'homme a eu très tôt besoin de se doter de systèmes de numérations. Mais comment construire un système de

numération suffisamment puissant pour prendre en compte l'infinité des quantités mesurables ? Comment construire un système de numération qui ait la puissance des nombres eux mêmes ? Après un long tâtonnement et des dizaines de systèmes de numération infructueux et limités, la solution est apparue dans le système de numération par position et l'invention du zéro. L'analyse que nous présentons au chapitre II débouche rapidement sur des conclusions similaires. Nous proposons un système de représentation qui encode des catégories dans des positions. Ce qui a empêché l'apparition d'un système de numération ayant la puissance des nombres eux-mêmes c'est le fait d'associer un sens aux chiffres. Comme la quantité de nombres potentiels est infinie, la quantité de chiffres est alors infinie. Les règles arithmétiques portant sur des systèmes de numération archaïques portent sur des nombres et ne sont qu'une juxtaposition de cas particuliers. Traiter le problème dans ce formalisme n'a pas de fin. Pour s'en convaincre, il suffit d'essayer d'exprimer 2^{64} en chiffre romain, ou de s'essayer de multiplier XIX par CIX. L'apport des systèmes de numération par position repose sur un nombre fini de chiffres, le sens des chiffres est alors uniquement fonction de leur position dans le nombre. Un tel système permet également d'exprimer n'importe quelle quantité et un ensemble fini de règles arithmétiques portant seulement sur les chiffres suffit alors à effectuer toutes les opérations possibles. L'infinité des sens est alors rejetée dans l'infinité des lieux. Ces règles sont locales et ne portent jamais sur le nombre entier.

L'analyse du problème conduit à un ensemble de propriétés. Notre approche considère des unités élémentaires de traitement admettant un ensemble d'états, dont l'état « zéro », et un ensemble fini de règles. Cet ensemble fini de règles ne porte que sur un voisinage d'unités de traitement et jamais sur l'ensemble du système. Il y a un parallèle évident avec les propriétés qui ont permis la construction du système de numération par position. Ces propriétés permettent de construire un grand nombre de systèmes. Nous proposons dans le chapitre III un modèle particulier. Ce modèle prend en compte l'ensemble des propriétés exhibées au chapitre II.

Une grande partie du travail de cette thèse à été consacré au développement de NeuSter qui est un simulateur du modèle proposé. Le modèle et le simulateur ont beaucoup évolué au cours des trois dernières années. Nous présentons au chapitre IV les grandes lignes du fonctionnement de NeuSter, en particulier les choix algorithmiques qui ont permis aujourd'hui de simuler des réseaux de plusieurs millions d'unités et milliards de connexions sur des clusters hétérogènes de machines POSIX.

Le chapitre V présente les premiers résultats obtenus avec NeuSter. Nous n'avons pas aujourd'hui la puissance de calcul suffisante pour fermer la boucle perception-action. Les premiers résultats portent essentiellement sur la construction d'un système de représentation ouvert utilisant la propriété de codage par position. Nous présentons à titre d'exemple des réseaux non supervisés catégorisant des orientations locales, des éléments géométriques de second ordre tels que des coins, des caractères d'imprimerie, des visages, des directions de mouvement et flux optiques et des phonèmes. Toutes ces expérimentations ont conduit à des convergences extrêmement rapides (le plus souvent moins d'une minute) et à des fréquences de fonctionnement de l'ordre de 10 à 15 Hz pour des réseaux de l'ordre de 50000 unités et quelques millions de connexions.

Plusieurs travaux relatifs aux idées proposées dans cette thèse ont été publiés [Paquier 02],

[Paquier 03a], [Paquier 03b]. Pour autant aucun d'eux ne fait explicitement référence à la notion de codage par position, cette notion y est implicite.

Chapitre I

Panorama des fonctionnalités, motivations et représentations en robotique

Notre travail repose sur une réflexion à propos de la notion de systèmes de représentation et de fonctionnalité ouverts. La classe de systèmes que nous allons étudier peut être définie comme n'ayant pas de représentation et de fonctionnalité préalablement fixées. Cette classe de système doit être capable de construire ses propres représentations et développer de nouvelles fonctionnalités de façon autonome. La conception de tels systèmes met en question les fondements mêmes de l'intelligence artificielle et doit peut-être conduire à l'élaboration de nouveaux paradigmes. Nous allons parcourir les travaux les plus représentatifs de la robotique actuelle à travers trois axes d'étude : la fonctionnalité de la machine, sa motivation, et le choix du système de représentation.

On peut considérer qu'il y a deux grandes branches en robotique. La première est une robotique industrielle qui est née dans les années 60 et qui se donne pour but d'automatiser les processus de production (peinture, soudure, saisie et déplacement). Cette approche de la robotique est issue de travaux en automatique et en mécanique. Elle est aujourd'hui présente dans tous les secteurs de l'industrie. Nous ne parlerons pas de cette robotique dans cette thèse. L'autre grande branche de la robotique est liée au mythe de l'intelligence humaine. Elle se base sur des travaux menés en intelligence artificielle. Contrairement à la première branche, cette robotique en est encore au stade de recherche. Les premiers travaux ont commencé dans les années 60. Les contraintes et l'imprédictibilité de l'environnement ont peu à peu amené l'introduction des notions d'adaptation et d'apprentissage, conduisant au développement de méthodes hybrides utilisant à la fois intelligence artificielle et apprentissage par renforcement, ou des approches connexionnistes. D'autres approches considèrent le robot comme un moyen d'étude. Le robot

n'est plus utilisé dans le but de développer des systèmes automatiques susceptibles de rendre un service à l'homme, mais plutôt pour valider ou appliquer des théories évolutionnistes ou développementales.

Le premier axe de notre étude repose sur la fonctionnalité de la machine. On définit les fonctionnalités d'un programme informatique par l'ensemble des possibilités que ce programme offre à un utilisateur ou à un autre programme. Par extension, nous parlons ici de la fonctionnalité des systèmes robotiques. Cette notion est fondamentale en robotique. La fonctionnalité des robots varie énormément d'un paradigme à l'autre. Les travaux inspirés de l'IA symbolique et les approches évolutionnistes abordent cette notion de façon complètement différente. Dans un cas il s'agit de proposer un service à un utilisateur final, dans l'autre il est question de la validation d'un modèle ou d'une théorie.

La motivation du robot peut être définie comme ce qui le pousse à agir. Cette notion est fortement liée à la notion de fonctionnalité. Elle peut être une commande donnée par un utilisateur ou être intégrée à la machine et inaccessible à une interaction avec un utilisateur. Là encore, cette notion est très dépendante du paradigme considéré.

La notion de représentation est sans doute la plus discutée à l'heure actuelle. Nous entendons par représentation, l'information que le robot a de son environnement, au sens le plus large. Il s'agit de savoir comment elle est acquise, traitée et stockée et réutilisée par le système. Selon les approches, le stockage ou le traitement de l'information peuvent être centralisés, ou distribués. Dans les approches symboliques, le traitement et le stockage de l'information sont séparés, alors que dans les approches connexionistes, ces deux aspects sont associés.

Peut-on envisager de développer un système d'apprentissage ouvert en s'appuyant sur les approches actuelles ? Permettent-elles de créer de nouvelles fonctionnalités et de nouvelles représentations sans intervention humaine ?

I.1 L'approche IA symbolique

Les différentes approches robotiques actuelles reposent directement ou indirectement sur les théories et méthodes issues de l'intelligence artificielle. Comprendre la façon dont les représentations et les fonctionnalités sont traitées en robotique aujourd'hui nécessite de s'intéresser aux origines de l'intelligence artificielle.

I.1.1 Fondements théoriques de la robotique, la naissance de l'IA

La majorité des sciences trouvent leur racine dans les réflexions des penseurs de la Grèce antique. L'intelligence artificielle et la robotique ne font pas exception à cette règle. Nous allons exposer ici les travaux qui ont permis le développement de ces deux sciences au XXème siècle.

Wiener introduit le terme de cybernétique [Wiener 43] et unifie les notions de communication, de contrôle et de mécanique statique, dans les machines et les systèmes vivants. Alan Turing peut également être considéré comme l'un des principaux initiateurs de ce qui deviendra l'intelligence artificielle. En 1950 paraît "computing machinery and intelligence" [Turing 70]. Cet article pose pour la première fois la question de la pensée des machines.

C'est également dans les années 40 que McCulloch et Pitts fondent la théorie des réseaux de neurones formels [McCulloch 43]. Ils montrent qu'un réseau de neurones formels est équivalent d'un point de vue computationnel à une machine de Turing. Chaque neurone est modélisé comme un automate à seuil dont l'état, actif ou non, désigne une valeur vraie ou fausse selon qu'il soit actif ou non. De tels neurones peuvent alors être interconnectés de façon à former des portes logiques. De cette façon, le cerveau tout entier peut être considéré comme une machine déductive.

Vers le début des années 50, des publications en psychologie, ingénierie et mathématiques convergent vers un thème commun : modéliser le comportement humain en situation de jeu. Le jeu d'échecs devient rapidement l'un des paradigmes phare de l'IA [Shannon 50]. En 1956, John McCarthy organise les conférences Darmouth, et pour la première fois apparaît le terme "intelligence artificielle". Il qualifie alors un champ d'étude regroupant cybernétique et théorie des automates. Un débat sur la possibilité qu'une *machine puisse penser* va alors naître. L'ambition de cette nouvelle techno-science est initialement immense [1] : permettre à l'ordinateur d'imiter le raisonnement humain avant la fin du siècle.

De toute évidence, la réponse à ces questions dépend fortement de la définition qu'on donne à *machine*, *pensée*, et *intelligence*. Les recherches sur la calculabilité et la reproductibilité de la pensée humaine ont pris leur essor dans la première partie du XXème siècle. Il faut cependant garder à l'esprit que les connaissances sur le fonctionnement du cerveau humain ont considérablement évolué au cours du siècle dernier. Toutes les théories traitant du fonctionnement de l'esprit humain ont suivi cette évolution. De fait, pour mieux comprendre la naissance d'un paradigme, il faut le replacer dans son contexte scientifique et technologique. À la fin du XIXème siècle, Pearson [Pearson 92] compare le fonctionnement du cerveau à celui d'un central téléphonique[2]. Plus tard, dans les années 50 on commence à faire le parallèle entre le cerveau et l'ordinateur. Dans les années 70, Pribram compare la mémoire humaine à un système holographique [Pribram 74]. Enfin, dans les années 80, avec l'essor de la théorie du chaos, Freeman [Karda 87] présente le cerveau comme un système dynamique.

Au début des années 40 on compare le fonctionnement du cerveau humain à celui d'une machine de Turing. Le sens donné à la *pensée*, avant même les conférences Darmouth est fortement inspiré de la théorie de Turing, en particulier le fonctionnement de sa fameuse machine. L'IA entreprend dès ses premières années de construire un modèle symbolique inspiré du raisonnement humain de haut niveau. Il s'agit ici de la manipulation de symboles à l'aide d'un jeu de règles appliquées séquentiellement. Newell et Simon ont grandement contribué au développement de ces théories [Newell 90], [Feigenbaum 63]. Newell propose "The Physical Systems Symbol Hypothesis", tandis que Simon formalise les implications de ce type d'approche. Le problème du sens est abordé par Minsky comme la forme des relations entre les

[1] L'une des premières conférences d'IA s'intitule : " Que ne peut l'informatique ?"

[2] Pearson, 1892, "*The view of brain activity here discussed may perhaps be elucidated by comparing the brain to the central office of a telephone exchange, from which radiate to the subscribers A, B, C, D, E, F, &c., who are senders, and to W, X, Y, Z, &c. who are receivers of the messages. A, having notified to the company that he never intends to correspond to anybody but W, his wire is joined to W, and the clerk remains unconscious of the arrival of the message from A and its dispatch to W, although it passes through his office.*"

symboles [Minsky 68].

Penser c'est alors *manipuler des symboles concrets ou signes physiques*. Harnard définit les systèmes symboliques comme un ensemble de signes distincts inscrits sur un papier, de trous sur une bande, ou d'événements dans un ordinateur, etc. qui sont manipulables sur la base de règles explicites elles-mêmes similaires à ces signes ou chaînes de signes. La manipulation de ces signes est seulement basée sur leur forme (pas leur sens), i.e, elle est purement syntaxique et consiste en une combinaison et recombinaison de ces signes. Il existe des signes primitifs et atomiques et des chaînes de ces signes. La syntaxe peut systématiquement être assignée à un sens [Harnad 90]. Dans cette définition Harnard passe sous silence la question du sens des symboles pris individuellement et le processus qui consiste à assigner un sens à un symbole. Nous verrons plus loin que cet aspect de la sémantique constitue l'un des problèmes fondamentaux de l'IA.

Le terme robot est popularisé par le dramaturge Tchèque Karel Capek[3] en 1921. Robot est un terme qui désigne des ouvriers ingénieux dans la pièce *R.U.R. : Rossums Universal Robots*. Le Robot Institute of America définit la robotique en 1979 comme ceci : "Un manipulateur reprogrammable et multifonctionnel, développé dans le but de déplacer des matériaux, des pièces, des outils ou des appareils spécialisés à l'aide de plusieurs programmes de déplacement et pour la réalisation d'un ensemble de tâches". Certains définissent alors la robotique comme la rencontre de l'IA et du monde réel.

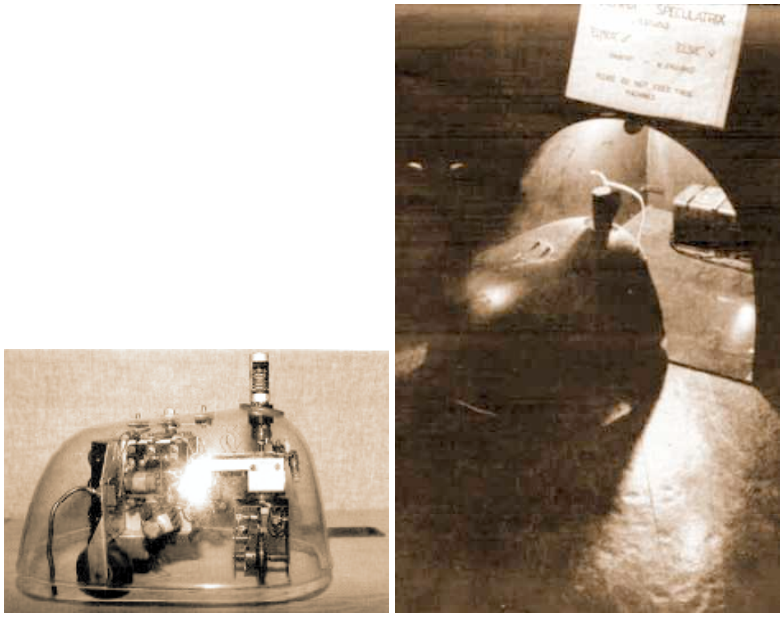


FIG. I.1 – Elsie, USA, 1950.

[3]Histoire de Karel Capek : http ://capek.misto.cz

Revenons sur les travaux fondateurs. En 1950, Grey Walter développe un robot "tortue" qu'il baptise Elsie. Ce robot se dirige vers les sources de lumière tant qu'il a suffisamment d'énergie puis revient vers sa source d'énergie. Ce comportement élémentaire est alors comparé à l'intelligence d'une bactérie. Toute l'intelligence de la machine est encore analogique, il n'y a toujours pas d'ordinateur embarqué. Ces travaux sont également considérés comme fondateurs de la vie artificielle (VA). En 1964, John Hopkins développe Beast. Ce robot utilise des sonars pour se diriger en intérieur. Son but est de trouver des prises de courant. En 1965, Raj Reddy fonde le Robotics Institute à Carnegie-Mellon University. A Stanford, Moravec développe en 1970 le premier robot d'extérieur. Le premier robot moderne est Shakey. Il a été développé au SRI de 1966 à 1972. Il est le premier robot mobile à tenir un raisonnement sur ses actions.

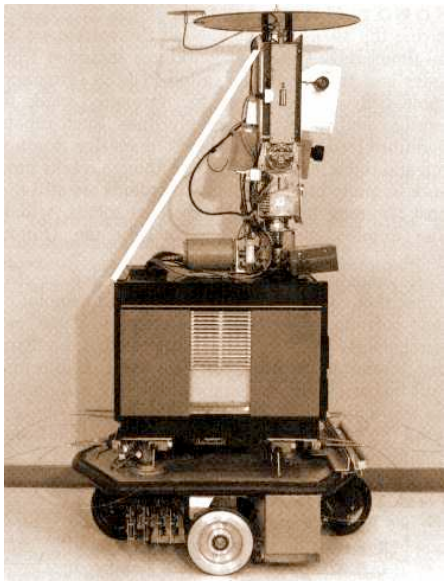


FIG. I.2 – Shakey, SRI, USA, 1966-1972.

On trouvait déjà sur ce robot : Une camera vidéo, des capteurs de choc, une connexion radio vers un DEC PDP-10 et PDP15 ainsi qu'un système de triangulation. Il possédait des programmes pour la perception de l'environnement, la modélisation du monde, et l'action. Les algorithmes bas niveau permettaient d'avancer, tourner, et planifier une trajectoire. Les algorithmes intermédiaires permettaient de combiner les algorithmes de bas niveau de façon plus robuste, enfin les algorithmes de haut niveau servaient à exécuter des plan pour atteindre des buts donnés par un utilisateur. Le robot Shakey a eu une influence majeure sur la robotique orientée IA d'aujourd'hui.

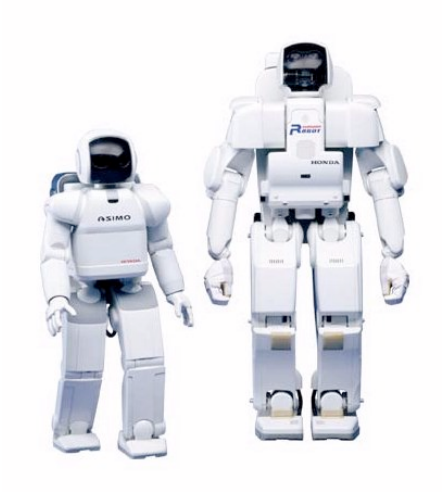


FIG. I.3 – Asimo, Honda, Japon, 2003.

I.1.2 Les approches purement symboliques

Les approches purement symboliques reposent le plus souvent sur une méthode identique basée sur la modélisation d'un ensemble de problèmes précis et leur mise en oeuvre. Concevoir un tel système commence par une analyse poussée du problème que l'on souhaite que le robot résolve.

Un robot est un système qui doit percevoir son environnement, et agir sur lui. Modéliser un ensemble de fonctionnalités nécessite le développement d'algorithmes de perception, de commande, de planification et de supervision. Durant la phase de modélisation, les concepteurs du système déterminent les représentations les plus adaptées aux fonctionnalités dont doit disposer la machine. Cette étape revient à extraire les paramètres essentiels à la résolution du problème. Les paramètres perceptifs sont explicités dans le programme et constituent les symboles manipulés par le planificateur et le superviseur. Les représentations de la machine et les règles d'inférences utilisées par le superviseur sont fixées pendant la phase de conception. Ce type de système est dédié à un ensemble de tâches fermé et ne peut être étendu en cours de fonctionnement.

La navigation autonome [Alami 98] et la manipulation constituent les deux branches principales de ce type d'approche. La navigation autonome pose explicitement le problème de la localisation dans l'environnement. Le robot doit connaître sa position pour se déplacer. La machine est conçue autour d'une fonctionnalité principale qui consiste à naviguer de façon autonome d'un point à un autre. Les représentations sont donc construites dans ce but. La perception de la machine consiste à se localiser dans une carte en utilisant les données issues de ses capteurs. Chaque type de capteur met en oeuvre une série d'algorithmes spécifiques, ce qui pose un problème de fusion des données. La carte utilisée par le robot pour se localiser est

donnée à l'avance ou construite en ligne (simultaneous localization and mapping, SLAM). La manipulation pose le même type de problème. D'une manière générale, les représentations que le robot a de son environnement sont déterminées pendant la phase de conception et constituent l'état du monde du robot. Les robots Asimo (Honda), même s'ils paraissent très avancés technologiquement, utilisent aujourd'hui ce type d'approche (voir figure I.3). Ces machines sont en mesure de monter des escaliers, marcher et se relever. Toutes ces fonctionnalités sont explicitement codées durant la phase de conception du système.

Selon cette approche, la motivation de la machine reflète la motivation de l'utilisateur. La conception de tels robots vise à proposer un ensemble de services à l'homme en automatisant des tâches préalablement fixées. Ces services constituent les fonctionnalités de la machine. Une fois le logiciel du robot mis en oeuvre, la machine n'a aucun moyen d'accroître ses représentations et ses fonctionnalités de façon autonome. Cette approche suffit dans certains cas, mais n'est pas adaptée à la conception d'un système de représentations et de fonctionnalités ouvertes. Elle a été très tôt critiquée.

I.1.3 Les critiques du symbolisme

L'approche symbolique de la robotique pose un grand nombre de problèmes, en particulier sur la façon de modéliser les changements du monde dûs à une action du robot. Ce problème est appelé "frame problem". Les approches symboliques ou cognitivistes constituent un problème d'encodage des connaissances pour les concepteurs. Comment prévoir les conséquences de toutes les actions en tenant compte de tous les contextes de l'environnement [Bickhard 95] ? Modéliser explicitement l'ensemble des cas possibles conduit à l'explosion combinatoire. Certaines conséquences ne peuvent pas être explicitement prévues par le concepteur.

Dès les années 70, Hubert Dreyfus, professeur de philosophie à Berkeley [Dreyfus 79], soulève un grand nombre de questions sur la possibilité qu'une intelligence artificielle puisse atteindre le niveau de l'intelligence humaine. Sa critique porte sur l'approche symbolique de l'IA. Il pose en effet une question essentielle : existe-t-il ou non, dans l'intelligence humaine, quelque chose qui puisse donner sérieusement à penser qu'aucune intelligence artificielle ne lui ressemblera jamais vraiment ?

Ce problème va réapparaître périodiquement sous des formes un peu différentes. Searle soulève le problème de l'ancrage au début des années 80. Il est déclenché par le "Chinese Room Problem" présenté dans l'article "Minds, Brains and Programs" [Searle 98] et repris par Harnard en 1990 [Harnad 90]. Selon l'approche classique de l'IA, la machine manipule des symboles sur la seule base de leur forme, le sens est attribué par l'utilisateur ou le concepteur du système. Les traitements sont uniquement syntaxiques. Les symboles sont connectés au monde à travers des traducteurs sensori-moteurs. Ces traducteurs ne sont en aucune manière une source de sens. Selon Harnad, le problème de l'ancrage peut être résolu par un système doté de capacités de discrimination et d'identification des entrées sensorielles.

Il existe également une critique phénoménologique de l'IA. La phénoménologie d'Heidegger propose des arguments contre la modélisation rationaliste des processus mentaux. Dans l'approche phénoménologique, toute action est ancrée dans une situation historique concrète.

Toutes les représentations sont dépendantes d'un contexte et il n'est pas possible d'en extraire des représentations logiques finies et indépendantes. La phénoménologie propose des critiques de l'approche symbolique, mais n'apporte pas de solution au problème. L'ensemble des critiques s à l'encontre d'un système purement symbolique et les difficultés de mise en oeuvre de tels systèmes ont peu à peu conduit à des approches hybrides mêlant symbolisme et apprentissage.

I.2 Les approches hybrides

Les robots perçoivent leur environnement à travers des données issues de leurs capteurs. Le monde est très riche et les situations rencontrées sont toujours différentes. De plus, toutes les situations rencontrées ne peuvent être explicitement prévues. Il est nécessaire que la machine s'adapte aux conditions réelles qu'elle rencontre pour parvenir à accomplir les tâches pour lesquelles elle a été conçue. Les approches probabilistes constituent un premier pas vers une adaptation aux situations réellement observées. Ces approches utilisent la théorie de Markov, en particulier la notion d'état et de transition d'état. Les méthodes les plus utilisées sont les MDP[4], ou les POMDP[5]. Elles ont par exemple été utilisées pour des problèmes de navigation autonome [Koenig 96] [Morisset 02] [Thrun 99]. L'apprentissage par renforcement et les approches connexionistes constituent les deux approches majeures dans l'intelligence artificielle adaptative. Il arrive souvent que les méthodes d'apprentissage par renforcement utilisent aussi des MDP ou POMDP.

I.2.1 Apprentissage par renforcement

L'apprentissage par renforcement a été introduit par Richard Sutton et Andrew Barto à l'université du Massachusetts [Sutton 98b] [Sutton 98a]. Cette approche consiste à faire en sorte que la machine apprenne des correspondances entre un ensemble d'actions et de situations dans le but de satisfaire un ensemble de critères relatifs à une récompense. Les idées relatives à cette approche sont issues de théories sur l'apprentissage par essai-erreur. En psychologie la notion d'apprentissage par renforcement est communément admise. Edward Thorndike [Thorndike 11] est un des premiers à proposer une théorie sur la notion d'apprentissage par essai-erreur chez l'animal. Il remarque que les actions qui précèdent une récompense ou une punition sont respectivement réutilisées dans les mêmes conditions ou abandonnées. Formellement un tel modèle consiste en un ensemble discret d'états de l'environnement, un ensemble fini d'action et un ensemble de signaux de renforcement, typiquement un réel dans l'intervalle $[0,1]$. L'agent doit trouver une politique π de mise en correspondance d'états et d'actions qui maximise le critère de renforcement [Kaelbling 96].

De nombreux travaux ont été menés dans ce domaine. Par exemple, la localisation quantitative d'un robot basé sur une représentation métrique pose de nombreux problèmes. Kuipers et

[4]MDP : Markov Decision Process

[5]MDP : Parcially Observed Markov Decision Process

Koenig ont proposé des méthodes de localisation qualitative dans une carte mettant en oeuvre des méthodes d'apprentissage par renforcement [Kuipers 94][Koenig 96]. Ces méthodes s'appuient à la fois sur des POMDP et des algorithmes de type $TD(\lambda)$ ou $Q-learning$. Plus récemment Chris Gaskett a utilisé ces approches pour l'apprentissage en ligne de cartes sensorimotrices où un robot humanoïde doit suivre visuellement une cible et apprendre à atteindre cette même cible avec sa main [Gaskett 99] [Gaskett 00] [Gaskett 03] (Voir figure I.4).

Ce type d'approche apporte plus de souplesse aux méthodes d'intelligence artificielle classique, mais ne résout pas le problème des représentations. Les représentations sont toujours données durant la phase de conception du système. Les fonctionnalités sont également données par la donnée des signaux de renforcement. Il s'agit en réalité d'une méthode d'optimisation de comportement attendu par le concepteur du système.

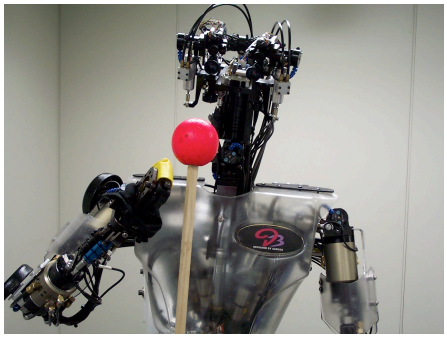


FIG. I.4 – Robot humanoïde, ATR, Japon, 2003.

I.2.2 Approches connexionistes

Les approches connexionistes constituent l'autre branche majeure de l'intelligence artificielle adaptative. Comme nous l'avons vu, les réseaux de neurones formels sont aussi anciens sinon plus que l'IA. Ils ont été étudiés dans les années 50 et 60, mais c'est au début des années 80 et avec l'arrivée d'ordinateurs suffisamment puissants qu'ils opèrent un retour important [Hopfield 82] [Kohonen 82]. Un réseau de neurones peut être entraîné de façon supervisée ou non supervisée. Les méthodes supervisées consistent à surdimensionner un problème difficile à modéliser. L'apprentissage du réseau permet d'approximer un ensemble de relations sans avoir à définir explicitement les paramètres essentiels à un phénomène à modéliser. Les réseaux supervisés sont également utilisés comme une méthode d'optimisation. Les réseaux de neurones non supervisés sont utilisés pour classifier des données. D'une manière générale, les réseaux de neurones formels sont rarement utilisés exclusivement. Il sont souvent utilisés en conjonction avec d'autres méthodes [Thrun 98][Wermter 00].

Les représentations dans un système connexioniste sont données par le concepteur dans le cas de systèmes supervisés. Les représentations sont fournies au système par l'intermédiaire

des bases d'apprentissage utilisées pour entraîner les réseaux. Les systèmes non supervisés ne sont pas capables d'associer seuls une étiquette à une catégorie. Pour cela il est nécessaire que le concepteur ou l'utilisateur réalise un liage, ce qui revient à créer des symboles pour le système. Sans ce liage, les algorithmes d'IA sont incapables de traiter des classes brutes indépendamment de leur sens. Très récemment, Paul Verschure a proposé un modèle robotique qui unifie l'IA classique et les approches connexionistes [Verschure 03]. Il existe des ponts entres les approches dites classiques de l'IA et les approches adaptatives.

La motivation d'un système d'IA adaptative est de même nature que celle des systèmes d'IA classique, elle n'est autre que celle du concepteur ou de l'utilisateur. Un tel système est construit pour proposer un service à un utilisateur. Ce type d'approche consiste à comprendre et modéliser un ensemble de propriétés. Une fois le problème modélisé, un logiciel spécifique est mis en oeuvre et testé sur un robot réel.

I.3 Les approches biologiquement inspirées

Agnès Guillot [Guillot 02] définit ce type de système comme un domaine de la cognition artificielle qui concerne dans un système non biologique (ordinateur ou robot) des processus permettant de construire, mémoriser, retrouver, transformer et transmettre des connaissances. Ce type d'approche considère le robot comme un moyen et plus comme une fin. Les robots développés dans ce cas sont aussi conçus autour d'un ensemble de représentations préalablement fixées. Les fonctionnalités sont apprises, mais la direction de cet apprentissage est également spécifiée implicitement. Contrairement aux approches classiques ou adaptatives, la motivation du robot fait partie intégrante du logiciel. Pour cette raison, ces systèmes ne sont pas facilement contrôlables. Il ne semble pas envisageable d'utiliser de tels systèmes sur un drone ou un robot supposé surveiller une installation à risque. Il s'agit d'une robotique expérimentale qui n'est pas orientée vers la conception d'un service proposé à l'utilisateur.

Il y a deux grandes familles : les approches phylogénétiques et épigénétiques. L'une s'appuie sur les théories évolutionnistes et s'inspire de l'évolution des espèces. L'adaptation du système est distribuée dans les générations successives. Au contraire, l'approche épigénétique considère le problème de l'apprentissage d'un agent unique. Ce dernier doit en théorie utiliser son expérience pour acquérir des représentations et des fonctionnalités. En pratique, aucun système de ce type n'a encore été conçu.

L'approche Animat introduite au début des années 90 par Jean Arcady Meyer et al. constitue l'une des principales branches de ce type d'approche [Meyer 91], [Meyer 00], [Hallam 02]. En toute rigueur, elle peut à la fois être considérée comme phylogénétique et épigénétique. En 1986 Rodney Brooks propose une architecture pour le contrôle de robot écartant toutes représentations cognitivistes [Brooks 86]. Selon lui, le robot doit être considéré comme un système entier évoluant dans un environnement naturel [Brooks 91b]. Il critique le paradigme cognitiviste en particulier les systèmes de représentation et de traitement de l'information [Brooks 91e], [Brooks 91c] [Brooks 90], [Brooks 92], [Brooks 98]. Les travaux de Brooks sont considérés comme la principale influence de l'approche Animat [Brooks 91a].

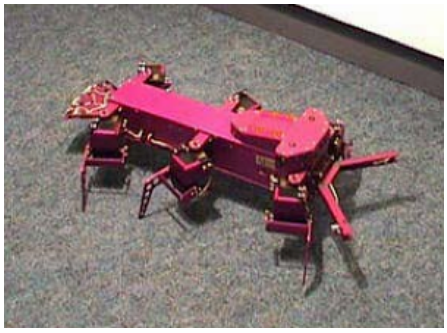


FIG. I.5 – HexaPod, AnimatLab, LIP6, France, 1998.

I.3.1 Les approches phylogénétiques

Les approches phylogénétiques en robotique s'appuient sur la théorie des algorithmes génétiques formalisés par John Holland dans les années 70 [Holland 75]. Le principe fondamental de ce type d'approche repose sur le concept de sélection naturelle élaboré par Charles Darwin. Selon cette approche, un robot n'apprend pas au cours de son expérience, mais une population d'agents possédant chacun des propriétés différentes est confrontée à un problème particulier et doit s'adapter [Nolfi 01], [Mondada, F. 95], [Floreano 94]. Un agent correspond alors à une solution potentielle. Chaque agent possède des gènes (les variables), des chromosomes, des parents et des descendants. Les principes d'évolution des populations sont basés sur des croisements d'agent ayant obtenu le meilleur score relativement à un critère donné, ou des mutations de code génétique. Ce type d'approche à en particulier été utilisé pour l'apprentissage de la marche d'un robot hexapode par l'équipe de Jean Arcady Meyer et Agnès Guillot (AnimatLab, voir figure I.5). Les représentations d'un agent dans ce cas sont les variables choisies pour construire le patrimoine génétique des populations étudiées (l'ensemble des gènes). Ces représentations sont liées à des propriétés du monde. La théorie des algorithmes génétiques est avant tout une méthode d'optimisation. Elle ne traite pas la question du choix des variables. Pour cette raison cette approche ne traite pas la question des représentations. La notion de fonctionnalité est ici liée aux critères que l'on choisit pour sélectionner les meilleurs individus. Si on souhaite apprendre des motifs de marche à un robot hexapode, on peut choisir la distance parcourue et la consommation d'énergie comme critère de survie. D'une manière générale la capacité à développer de nouvelles fonctionnalités dépend fortement de la capacité à construire de nouvelles représentations. Cette approche n'apporte pas de réponse sur la façon de représenter les informations relatives à l'environnement, nous devrons donc chercher dans une autre direction.

I.3.2 Les approches épigénétiques

Les approches épigénétiques considèrent le problème de l'apprentissage d'un agent à travers ses expériences. On peut classer dans cette famille d'approches la plupart des systèmes biologiquement inspirés utilisant en particulier des connaissances en neurosciences ou psychologie expérimentale [Gergely 03] [Steels 01]. Les travaux les plus récents utilisent des modèles de neurones à impulsion et des poids synaptiques dits "sparse". Certains travaux visent à construire des systèmes visuels artificiels se basant sur des données récentes en neurosciences computationnelles [Yang 03]. La robotique développementale s'inscrit dans cette approche. Très récemment, Christopher Prince, Luc Berthouze, Christian Balkenius et d'autres ont proposé de rassembler les différents travaux de robotique epigénétique et ainsi de matérialiser un nouveau champ de recherche. La robotique développementale est définie comme un champ de recherche pluridisciplinaire visant à intégrer psychologie développementale et robotique. Cette branche de la robotique est encore très jeune. On y trouve des travaux sur l'architecture de systèmes robotiques [Grupen 03], des travaux sur la perception, en particulier la vision active et la segmentation [Arsenio 03] [Lungarella 03]. Les robots utilisés sont souvent anthropomorphes. Le plus connu est sans doute Cog, conçu au MIT par l'équipe de Rodney Brooks (Voir figure I.6, gauche) [Brooks 91d] et BabyBot, un robot européen (LIRA-Lab, Gênes) relativement similaire à Cog (Voir figure I.6, droite). Il a, en particulier, été utilisé par Max Lungarella et Giorgio Metta. L'apprentissage par imitation est aussi une composante importante de ce type d'approche, en particulier l'acquisition du langage [Breidegard 03]. L'acquisition du langage est également traitée en conjonction avec des modèles de vision et d'audition [Zhang 03].

Aujourd'hui il n'existe pas véritablement de cadre de travail unifié pour ce type d'approche, il s'agit plus d'une direction de recherche. Il existe beaucoup de travaux sur des sujets très précis (vision, imitation, langage, manipulation), mais il est difficile de faire cohabiter ces différents modèles [Brooks 94], [Maes 90b], [Maes 90a].

La barrière à la conception de ce type de système est toujours la même. Sans un système de représentation ouvert, la machine reste dépendante de son concepteur. Si une machine doit apprendre au travers de son expérience, elle a besoin de stocker et de retrouver des informations de façon autonome. La notion de système ouvert, ou « open-ended » commence à prendre une place importante dans ce domaine de la robotique et de l'IA en général [Steels 95], [Kaplan 03], [Zhang 03]. Luc Steels a beaucoup travaillé sur l'origine du langage [Steels 97b], [Steels 97a], [Steels 98], [Steels 02]. Ce type de questions s'attaque directement au problème de l'ancrage des symboles (les mots)[Steels 96]. Apprendre de nouveaux mots et leur donner un sens nécessite de disposer d'un système ouvert de représentation. C'est dans cette direction que notre travail s'inscrit. Nous proposons dans cette thèse un système de représentation ouvert suffisamment puissant pour permettre d'encoder de nouvelles informations de façon autonome, et de les utiliser pour la construction de nouvelles fonctionnalités.

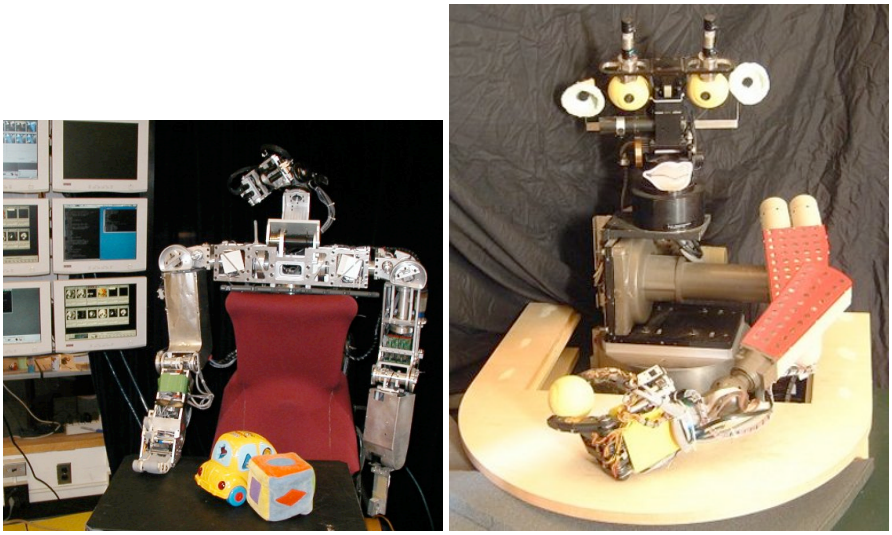


FIG. I.6 – (Gauche) Cog, IA-LAB, MIT, USA, 2003. (Droite) BabyBot, LIRA, Italie, 2003.

I.4 Conclusion

Dans ce chapitre nous avons vu qu'il existe deux écoles radicalement différentes en robotique. La première est une branche de l'IA et a pour objet de construire des machines intelligentes proposant un ensemble de fonctionnalités comme la navigation mobile, la manipulation ou l'interaction homme-machine. Ces fonctionnalités peuvent être considérées comme des services proposés à un utilisateur. Cette approche nécessite que les fonctionnalités soient explicitement implémentées sur le robot. Ce type d'approche ne permet pas qu'une fonctionnalité se développe durant la phase de fonctionnement. Les approches hybrides ajoutent des propriétés adaptatives aux approches classiques, mais ne touchent pas à la notion de fonctionnalité. L'adaptation porte sur l'optimisation des fonctionnalités, et non pas sur leur développement. Elles ne sont donc pas adaptées à notre approche.

L'autre école utilise des robots afin de tester des théories évolutionnistes ou développementales. Le robot n'est plus une finalité, mais il devient un moyen pour comprendre ou modéliser. Les théories évolutionnistes intègrent la notion de fonctionnalité dans le critère de survie, et les individus les mieux adaptés à la fonctionnalité recherchée sont croisés entre eux. Il s'agit également de l'optimisation d'une fonctionnalité. Seules les approches épigénétiques et développementales qui sont encore très jeunes s'intéressent à ces questions, mais il n'y a aujourd'hui aucune théorie au sujet de l'apprentissage ouvert de comportements. Le plus souvent les travaux concernant la robotique épigénétique ou développementale traitent seulement une partie du problème. Les différents aspects du problème sont pensés dans un formalisme qui leur est propre. Ils ne sont pas conçus en partant de considérations très générales sur la notion de représentation ou de fonctionnalité, mais traitent chacun une partie spécifique du

problème. Ces différents modèles ne sont pas pensés pour cohabiter au sein d'un seul système. Comment faire cohabiter plusieurs des systèmes de représentation très différents ? Comment extraire des propriétés générales de ces travaux ? Est-il possible d'utiliser de telles approches pour développer notre système ? Nous proposons de considérer la question des systèmes de représentation et de fonctionnalité de la manière la plus générale possible. Le modèle proposé ne doit pas être spécifique à un aspect particulier des fonctionnalités potentielles de l'agent.

Chapitre II

Apprentissage ouvert, analyse du problème

L'homme possède des capacités d'apprentissage et d'adaptation uniques. Il naît avec un ensemble de besoins physiologiques, mais n'a aucune connaissance *a priori* sur le monde qui l'entoure et les problèmes qu'il devra surmonter. Il va sans cesse accroître sa capacité à se représenter le monde et va développer des comportements de plus en plus complexes et adaptés à son environnement. Il va apprendre en expérimentant son environnement, en généralisant des situations, et va partager ses expériences en communiquant. Cet apprentissage est toujours incrémental. La propriété essentielle semble être la capacité à généraliser. De cette façon, une situation inconnue peut toujours être ramenée à un ensemble d'éléments principaux avec lesquels un comportement adapté a déjà été développé. L'individu commence par maîtriser la direction de son regard et développer ses capacités de mouvement et de manipulation. Il va apprendre la constance des objets, le fait que deux objets ne peuvent pas coïncider. Petit à petit il apprend à communiquer et à désigner les objets qui l'entourent. Plus tard, il attribuera des connaissances aux autres et pourra ainsi se mettre à leur place. Parallèlement, il apprendra à utiliser des outils ou à créer de nouveaux objets. La culture fait partie intégrante de l'environnement de l'individu. Elle permet à l'homme d'acquérir rapidement un ensemble de connaissances et des comportements très riches. C'est en cherchant à assouvir ses besoins que l'homme accroît ses connaissances et développe de nouveaux comportements.

Ce que nous voulons comprendre et modéliser, c'est précisément cette capacité d'adaptation au monde, en partant d'un système le plus simple possible, pour faire en sorte qu'au cours de son développement ce système puisse accroître sa capacité à se représenter son environnement et qu'il puisse développer de nouvelles fonctionnalités. Nous ne souhaitons pas imiter l'homme ou avoir une approche biologiquement inspirée. Nous voulons comprendre comment un système d'apprentissage ouvert doué de capacité de généralisation peut lui même

développer un ensemble de représentations et de fonctionnalités. Comment un système de ce type peut se donner des buts qualitativement différents en poursuivant un objectif unique ? Notre approche repose sur l'analyse d'un tel système. Nous proposons dans ce chapitre un ensemble de conditions parfois nécessaires parfois suffisantes qui vont nous conduire à la synthèse d'un modèle de ce type d'agent.

Comme nous l'avons vu dans le chapitre précédent, les robots sont toujours développés en fonction des buts que l'on souhaite leur faire atteindre ou des comportements que l'on souhaite voir émerger : ils ne sont pas capables de créer de nouveaux comportements ou de nouvelles fonctionnalités par eux-mêmes. L'ensemble des opérations que ce type de système sera capable d'effectuer est fixé durant la phase de conception. L'apprentissage est alors une optimisation de tâches définies initialement. En cela notre approche se fixe un but radicalement différent de celui de la majorité des approches actuelles.

La notion d'apprentissage ouvert soulève un grand nombre de questions, qui concerne en particulier la façon d'apprendre à représenter l'environnement sans connaissance *a priori* et la façon de construire des comportements adaptés en utilisant ces représentations. Quel peut être l'objectif ou l'intentionnalité de tels systèmes ? Nous commencerons par analyser les différentes contraintes que suppose une telle approche, en partant de considérations très générales. Après une analyse des propriétés nécessaires à un tel système, nous pourrons fixer les grandes lignes de notre modèle, puis nous terminerons en proposant une architecture générale qui puisse satisfaire l'ensemble des contraintes que nous allons mettre à jour.

II.1 Quelques définitions

Il est important de réduire au maximum les ambiguïtés de compréhension relatives à notre approche, nous allons donc commencer par définir les notions de motivation, d'objectif, de but, de fonctionnalité, de performance et de connaissance. Penchons-nous sur ces notions à travers un ensemble d'exemples empruntés aux principaux paradigmes de la robotique actuelle.

Commençons par définir les notions citées plus haut dans le cas d'un système de navigation autonome. Considérons un robot mobile disposant de capacités de localisation et de planification de trajectoire. Supposons que la tâche $ALLER\_A(X,Y)$ soit implémentée sur ce robot et qu'un utilisateur lui donne l'ordre $ALLER\_A(x_0,y_0)$. Dans ce cas, le but pour le système sera d'atteindre une position (x_0,y_0), c'est-à-dire de faire en sorte que son état interne vérifie au moins la condition $(X = x_0, Y = y_0)$. La fonctionnalité est la capacité à exécuter l'ordre $ALLER\_A$, c'est une possibilité offerte par le système. La connaissance du système est sa localisation, et la localisation de son but, c'est-à-dire (x,y), et (x_0,y_0). Dans ce cas précis l'objectif du système est d'exécuter les ordres que donne l'utilisateur.

La **connaissance** est l'information dont le système dispose sur le monde et sur son propre état.

Un **but** est un état à atteindre : atteindre un but, c'est agir pour passer d'un état courant à un état final particulier. Cet état final sera appelé état but.

Une **motivation** est un lien entre un acte et les raisons à l'origine de cet acte.

Nous définissons l'**objectif** du système comme la raison à l'origine de ses actes. Il s'agit d'un but de niveau qualitativement supérieur. Un but est un état particulier que le système cherche à atteindre. L'objectif est la cause, le "pourquoi" de l'existence de ces buts du système. Il faut faire attention à ne pas confondre l'objectif du concepteur et l'objectif du système. Ces deux objectifs peuvent être complètement différents.

Une **fonctionnalité** est un moyen d'action. Du point de vue d'un observateur, aller quelque part, prendre quelque chose, reconnaître une chose, manipuler une chose, utiliser une chose, construire un outil, communiquer constituent des fonctionnalités clairement identifiables. Chaque fonctionnalité se différencie des autres parce qu'elle est qualitativement différente et clairement identifiable.

La **performance** est l'aspect quantitatif de la fonctionnalité. Chaque fonctionnalité est réalisée avec une performance. Il doit être possible de comparer deux systèmes qui possèdent la même fonctionnalité, à travers leur performance. Il y a plusieurs façons d'implémenter la tâche *ALLER_A*, cependant, selon certains critères, certaines auront de meilleures performances que d'autres.

Par exemple, considérons un robot à pattes qui apprend des motifs de marche efficaces à l'aide de méthodes évolutionnistes. Ce robot va acquérir la fonctionnalité de la marche, et améliorera sa compétence au cours de l'évolution de son espèce. L'objectif de l'espèce est de marcher le mieux possible. Le but de chaque individu est de parcourir la distance maximale en consommant le moins d'énergie. La connaissance de chaque individu est constituée de la position des pattes, de la distance parcourue, de l'énergie dépensée, et de la dernière séquence de mouvement effectuée.

Les propriétés que nous souhaitons donner à la machine sont la capacité d'acquérir de nouvelles connaissances, d'apprendre de nouvelles fonctionnalités, et d'améliorer la performance associée à chaque fonctionnalité. Dans cet exemple précis, nous souhaitons qu'au cours de son apprentissage, la machine puisse proposer la fonctionnalité *ALLER_A* alors qu'elle n'en disposait pas initialement et sans qu'aucun programme relatif à cette fonctionnalité ne soit écrit par les concepteurs du système. Nous voulons qu'elle apprenne de nouvelles connaissances (ici, des connaissances concernant les lieux), et qu'elle accroisse ses compétences en réalisant ses fonctionnalités.

Nous utiliserons souvent le terme de situation. Il englobe ici l'état de l'environnement ainsi que l'état du système lui-même.

Se pose alors la question des buts et de l'objectif d'une telle machine.

II.2 Objectifs

Nous distinguons deux objectifs. Le nôtre, qui consiste à développer un système capable d'augmenter ses fonctionnalités, et l'objectif propre au système. Nous faisons l'hypothèse que le système que nous cherchons à développer est un agent hédoniste. Chaque situation devra potentiellement avoir un effet sur lui, c'est à dire lui procurer du plaisir, ou de la douleur. Nous appellerons effet positif le plaisir et effet négatif la douleur. Ces notions seront précisées plus

loin. Le système va chercher à agir pour se trouver dans des situations qui vont lui procurer un effet positif et éviter les situations qui vont lui procurer un effet négatif. Ces situations sont imposées initialement, et non choisies par le système.

Si ce dernier pouvait choisir les situations associées à un effet positif, il n'aurait aucun besoin d'apprendre. Il pourrait par exemple choisir la première situation rencontrée, lui attribuer une connotation positive et ne plus jamais avoir à agir. L'objectif du système sera d'accroître la perception d'effets positifs et en corollaire de décroître la perception d'effets négatifs. La contrainte porte sur un accroissement et non pas sur la recherche d'un maximum, ainsi le système ne se contentera pas d'une situation associée à l'obtention d'un effet positif, mais cherchera toujours une situation associée à un effet positif meilleur.

Notre analyse sera centrée sur le point de vue du système. Nous procédons ainsi à un changement de point de vue qui va conduire à un changement d'objectif. Alors que pour nous, les propriétés qui consistent à l'accroissement des capacités à représenter l'environnement et à agir sont des finalités que nous souhaitons donner au système, pour lui, elles sont un moyen d'atteindre sa propre finalité.

Nous pouvons montrer par l'absurde que le fait que le système atteigne son objectif implique le fait que nous atteignions le nôtre. Supposons que nous disposions d'un système qui cherche à accroître l'effet positif perçu et qui ne connaît initialement rien sur son environnement. S'il agit sans connaître son environnement, il n'aura pratiquement aucune chance de produire des actions appropriées qui vont conduire aux situations connotées positivement. Il n'aura donc pratiquement aucune chance de se trouver dans une situation connotée positivement, et ainsi d'accroître sa perception d'effets positifs. Sans connaissance suffisante sur son environnement, il ne parviendra pas à atteindre son objectif. De la même façon, il ne saura pas éviter les situations connotées négativement. C'est en cherchant à maximiser les situations connotées positivement que le système va devoir accroître la connaissance qu'il a de son environnement et cette connaissance lui servira à apprendre à produire des actions qualitativement et quantitativement de plus en plus appropriées.

Le fait que le système atteigne son objectif implique que nous atteignions le nôtre. L'objectif du système va demeurer constant au cours de son évolution alors que les comportements qu'il va développer pour l'atteindre vont s'enrichir. Nous allons donc analyser les propriétés d'un agent hédoniste qui cherche à accroître sa perception d'effets positifs.

II.3 Hypothèses de travail

Supposons que nous disposions d'un système robotique que nous appellerons également agent. Nous avons défini l'objectif de l'agent, nous allons maintenant présenter les différentes structures qu'il doit posséder ainsi que leurs propriétés les plus générales. Une fois que nous aurons présenté la classe de systèmes que nous souhaitons étudier, nous allons analyser l'effet de chaque condition et contrainte que l'objectif de l'agent exerce sur son architecture. Nous allons petit à petit détailler les propriétés nécessaires qu'une telle classe de système doit nécessairement posséder. Certaines de ces propriétés seront seulement suffisantes. Un résumé

des hypothèses est présenté figure II.1.

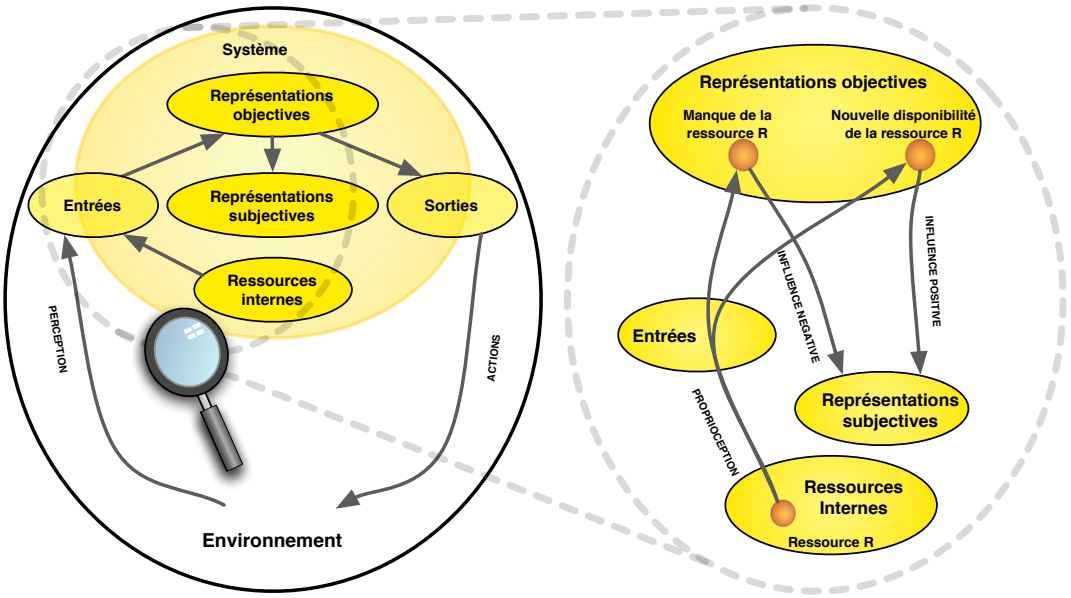


FIG. II.1 – Hypothèse de travail.

II.3.1 Situations de l'environnement et états internes de l'agent

Nous définissons une situation de l'environnement comme l'état de l'environnement à un instant donné. De la même façon, l'état interne de l'agent est constitué de l'état de l'ensemble des paramètres qui définissent ses différents sous-systèmes à un moment donné. Lorsque l'agent perçoit une situation, son état interne se trouve modifié. Lorsque l'agent agit sur son environnement il change de situation et par suite se trouve dans un nouvel état. L'agent n'a initialement aucune représentation de son environnement. La perception d'une situation a donc initialement peu d'influence sur son état interne.

II.3.2 Hypothèse hédoniste : objectif global

Nous supposons que notre agent est hédoniste, autrement dit son objectif est de rechercher son plaisir. Nous devons donc définir la notion de plaisir pour ce type de système robotique et mettre en place un mécanisme qui traduise cette propriété.

Chaque situation que l'agent rencontre contient des données objectives qu'il peut extraire comme les formes, les couleurs, les textures, etc. Chacune de ces perceptions objectives est associée à une sensation subjective. Nous appelons "effet subjectif" ces sensations subjectives. Ce type d'effet est subjectif dans le sens où il ne peut pas le mesurer avec ses capteurs. Ce

n'est pas une grandeur qui appartient au monde, mais une propriété qu'il attribue lui-même à ses perceptions. Nous modélisons cet effet par un nombre réel, c'est en quelque sorte un score que l'agent attribue à ces perceptions. Certaines situations produiront un effet subjectif positif, d'autres un effet subjectif négatif. Nous pouvons définir ainsi le plaisir de l'agent comme l'accroissement de l'effet subjectif qu'il perçoit. L'objectif de l'agent va consister à se trouver dans des états internes qui vont accroître son effet subjectif perçu, c'est à dire qui vont lui procurer du plaisir. La valeur de l'effet subjectif attribué aux perceptions objectives va évoluer au cours de l'apprentissage. Nous verrons que l'évolution de l'effet subjectif constitue le moteur de l'apprentissage des fonctionnalités.

II.3.3 Perceptions et Actions

Nous supposons que l'agent auquel nous nous intéressons est à la fois en mesure d'observer et d'agir sur le monde. Il possède des entrées et des sorties. Ses entrées sont des données acquises par ses capteurs y compris sur lui-même. L'agent possède des capteurs de proprioception. Son objectif consiste à se trouver dans des états internes qui lui procure du plaisir. Pour cela il peut agir pour changer de situation, ou simplement agir sur lui-même pour changer d'état interne. L'agent doit donc posséder des actions qui soient en mesure de changer son état interne, en particulier ses représentations, sans modifier l'environnement. Initialement, ces actions particulières ne sont pas en mesure d'accroître sa sensation d'effet positif, car les représentations du système ne sont pas encore formées. Ses sorties sont donc des actions sur l'environnement réalisées par ses effecteurs ou des actions sur son propre état interne.

II.3.4 Ressources internes

Le système que nous étudions est situé et incarné. Il possède des ressources internes, c'est à dire un ensemble de ressources matérielles qui caractérisent une partie de son état interne à un moment donné. Ces ressources peuvent être son niveau d'énergie, le fait que des composantes matérielles soient en panne, etc. Le système possède également des capteurs liés à ses ressources.

Nous faisons une hypothèse particulière sur la connaissance du système de ses propres ressources internes. Nous supposons que le système est en mesure de savoir, pour chaque ressource, si elle lui fait défaut ou si elle est à nouveau disponible. Si on reprend l'exemple du niveau d'énergie, le système doit être capable de représenter le fait que le niveau d'énergie de sa batterie est trop faible et le fait que la batterie vient d'être rechargée.

Nous supposons que le système associe le manque d'une ressource avec une diminution de l'effet subjectif, et qu'il associe le fait qu'elle soit à nouveau disponible avec un accroissement de l'effet subjectif *i.e* lui procure du plaisir.

II.3.5 Représentation et traitement de l'information

Pour agir de façon appropriée, le système a besoin d'informations sur son environnement et sur lui-même. Nous appelons représentations ces informations et la façon dont elles sont organisées. Nous considérons la notion de représentation la plus générale possible, nous ne faisons pas d'hypothèse *a priori* sur la façon dont ces informations doivent être organisées ni sur la façon dont elles doivent être traitées.

Initialement, l'agent n'a pas de connaissance *a priori* sur son environnement. Il n'a accès au monde qu'à travers les données brutes de ses capteurs, il devra donc extraire et organiser les informations que vont lui fournir ses capteurs. Ces processus d'acquisition de connaissances devront être réalisés de façon autonome.

Nous supposons par contre qu'il sait déjà traiter et organiser les informations relatives à ses ressources internes.

Initialement, l'agent n'a pas de représentations de son environnement. Nous faisons donc l'hypothèse qu'initialement, aucune situation de l'environnement n'a d'influence sur les variations de l'effet subjectif qu'il perçoit.

Enfin, nous supposons que la quantité de mémoire de l'agent est bornée et qu'il n'est pas possible d'étendre cette capacité de stockage de l'information. De la même façon, nous supposons que les capacités de traitement sont bornées.

II.3.6 Buts initiaux

D'une manière générale, les buts de l'agent consistent à atteindre des états internes qui seront en mesure de lui procurer du plaisir. Ces buts sont des conséquences de son objectif global (Voir hypothèse II.3.4). Il arrivera souvent que l'agent rencontre des situations où cet objectif n'est pas réalisable. Initialement, il n'a pas de connaissances sur son environnement, il ne peut donc pas s'appuyer sur ses représentations pour atteindre son objectif. Ces buts initiaux sont liés au choix des ressources internes. Périodiquement, des contraintes matérielles liées à son caractère incarné vont provoquer des variations de l'état de ses ressources internes ce qui va conduire à une diminution de la perception d'effets subjectifs. A la suite de cette diminution, son objectif deviendra réalisable. Ses buts initiaux consistent alors à agir pour que ses ressources propres retrouvent leur niveau normal, cette situation lui procurera du plaisir *i.e.* lui permettra d'atteindre son objectif.

II.3.7 Fonctionnalités

En agissant, l'agent passe d'un état interne à un autre, soit en changeant de situation de l'environnement soit en changeant directement son état interne. Chaque état interne est associé à un état subjectif. Les fonctionnalités de l'agent sont les actions appropriées qui lui permettent d'atteindre son objectif. C'est à dire des actions qui lui permettent de se trouver dans des états internes qui lui procurent du plaisir. Nous verrons au cours de ce chapitre que le système de fonctionnalités se base sur le système de représentation.

L'agent que nous cherchons à caractériser devra apprendre à extraire des données objectives de son environnement (les représentations). Il devra également apprendre à associer des valeurs subjectives à ces représentations. De cette façon, il sera en mesure de diversifier ses fonctionnalités et d'être de plus en plus adapté à son environnement.

L'évolution des fonctionnalités de l'agent dépend de deux choses : le choix des ressources internes qui sont à l'origine des premiers buts et l'histoire du système dans son environnement. Les fonctionnalités que le système va potentiellement développer sont donc liées à son origine et à ses expériences.

II.3.8 Interaction et autonomie

Nous supposons que notre agent est autonome, c'est-à-dire capable d'agir sans aucune intervention extérieure directe. De son point de vue, l'expérimentateur fait partie de l'environnement. La seule interaction possible entre l'expérimentateur et l'agent doit passer par les entrées et les sorties du système. Il est possible que l'expérimentateur participe à l'apprentissage du système en le récompensant ou en le punissant. C'est-à-dire en modifiant l'état des ressources internes du système.

II.4 Vers un système ouvert de représentations

Nous n'avons jusqu'à présent fait aucune hypothèse sur la façon dont le système doit organiser et traiter l'information qu'il observe. Nous allons analyser comment le système peut représenter le monde à travers les différentes contraintes qu'impose notre approche. Comme nous l'avons vu dans la section II.3.5, les représentations de l'agent sont les informations objectives dont le système dispose pour agir. Ces informations peuvent dans un premier temps être grossièrement découpées en deux catégories. Les informations relatives à la perception de l'environnement et celles relatives à la production des actions.

La perception est fortement liée à l'action. Notre agent n'a initialement aucune information sur le monde, et ne possède pas d'actions adaptées aux situations qu'il rencontre. Il ne perçoit le monde qu'à travers des données acquises par des capteurs. A chaque instant, chacun de ses capteurs propose une mesure de grandeur physique du monde. Ces mesures sont organisées dans l'espace et forment des images, nous verrons plus loin que cette propriété est essentielle.

Apprendre à agir de façon appropriée à une situation c'est apprendre à trouver les éléments perceptifs de cette situation qui sont déterminants pour le choix ou la synthèse de cette action.

Nous écartons immédiatement les approches symboliques : elles présupposent que le système possède initialement des connaissances sur le monde et implémente des algorithmes de reconnaissance de forme spécifiques à ces formes. Notre approche interdit d'utiliser une grande partie des techniques et méthodes utilisées en reconnaissance des formes aujourd'hui, en particulier celles qui présupposent l'existence d'un symbole ou d'une forme à reconnaître dans les signaux d'entrée. Notre objectif est que le système apprenne les connaissances qui lui sont nécessaires.

L'idée la plus naïve d'action adaptée serait d'associer chaque image perceptive avec une action appropriée. Cette approche est à écarter car elle conduit rapidement à une explosion combinatoire du nombre d'associations et nous avons posé comme hypothèse que le système possède une quantité finie de ressources computationnelles.

II.4.1 Généralisation et Catégorisation

La complexité du monde est telle que le système ne rencontrera jamais deux fois la même situation, c'est à dire qu'il n'aura jamais deux fois les mêmes données sur ses capteurs. Pour agir de façon appropriée à des situations inconnues mais proches de celles qu'il a déjà rencontrées, il doit nécessairement posséder des capacités de généralisation.

Le système doit être en mesure de trouver des propriétés invariantes entre les différentes images perceptives qu'il rencontre. Trouver un point commun entre plusieurs perceptions proches relève de la catégorisation. Agir de façon appropriée c'est déjà agir en fonction de catégories perceptives.

Supposons que l'agent se retrouve face à deux situations extrêmement proches. Si l'agent doit agir différemment face à chacune de ces situations alors il doit percevoir la différence entre ces deux situations. Il doit à la fois savoir en quoi elles se ressemblent et en quoi elles diffèrent. Le système de représentation doit refléter cette propriété. La capacité à représenter la différence entre deux situations proches constitue le grain perceptif de l'agent en dessous duquel il ne possède pas suffisamment d'information pour savoir qu'une situation est différente. Il n'est pas nécessaire de posséder des propriétés de reconnaissance, seule la capacité de catégorisation est nécessaire.

II.4.2 Représentations distribuées

Essayons d'étendre notre hypothèse naïve. Est-il suffisant d'associer une seule catégorie par situation ? Si le système doit généraliser, il doit percevoir simultanément plusieurs catégories.

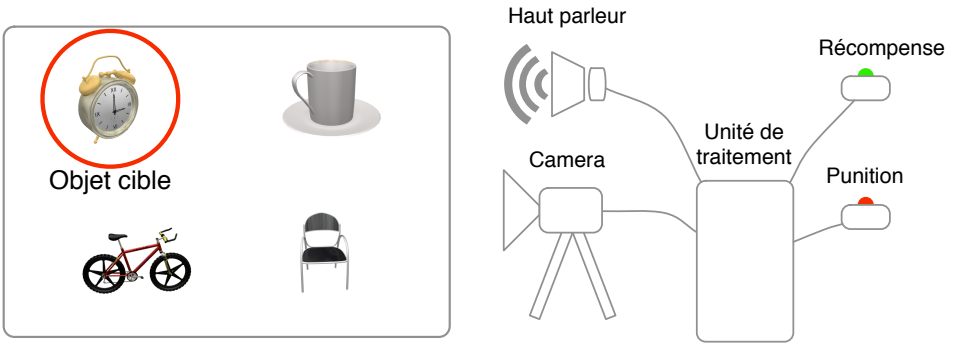


FIG. II.2 – Dispositif expérimental minimal.

	Action = Son	Action = Silence
Cible présente	Récompense	Punition
Cible absente	Punition	-

TAB. II.1 – Politique de récompenses et de punitions.

Pour montrer cette propriété, imaginons l'expérience suivante : Nous supposons que nous avons un agent muni d'une caméra, d'un haut-parleur et d'une paire de boutons dont l'un sert à le sanctionner positivement et l'autre négativement. Il possède une seule action qui consiste à produire un son. Initialement il ne connaît rien de son environnement, il répond[1] donc aléatoirement. Le dispositif expérimental est présenté figure II.2. Un expérimentateur choisit un objet cible parmi plusieurs (ici l'objet cible est un réveil). Les objets sont de taille fixe et toujours présentés sous le même angle. La cible est donc la forme d'un objet vu sous un angle particulier. Définir une cible de cette façon permet de simplifier le problème. Nous ne présentons ces formes qu'à deux positions possibles : l'expérimentateur présente à l'agent des combinaisons de formes qui contiennent ou non la forme cible. Du point de vue de l'expérimentateur, la tâche de l'agent consiste à trouver cette forme, c'est à dire produire un son quand la cible est présente dans l'image observée. Pour cela l'expérimentateur récompense l'agent si l'agent produit un son quand la forme est présente dans la scène qu'il observe, et le punit s'il produit un son quand la forme n'est pas présente ou s'il ne produit pas de son alors que la forme est présente. La politique de récompense et de punition est présentée Tableau II.1.

L'objectif de l'agent est d'accroître le nombre de récompenses qu'il reçoit. Pour cela, il doit caractériser l'information relative à la forme en question dans les scènes qui lui sont présentées. L'information nécessaire à l'adaptation de l'action se trouve dans la cause de la récompense et de la punition, c'est à dire dans la cause de l'accroissement ou de la diminution de plaisir. Si le système se contente d'associer une seule catégorie par scène rencontrée il n'aura pas de capacités de généralisation.

La figure II.3 montre quelques situations possibles. Les catégories détectées sont représentées par des disques blancs, celles qui ne sont pas détectées sont représentées par des disques noirs. Si une catégorie unique caractérise chaque situation, il est possible d'indexer les scènes en les numérotant. Supposons que pendant son apprentissage on présente seulement les situations de 1 à 5, et que l'agent apprenne parfaitement à agir dans ces situations. L'agent sera donc en mesure d'associer $(1-\mathit{Silence})$, $(2-\mathit{Silence})$, $(3-\mathit{Silence})$, $(4-\mathit{Son})$, et $(5-\mathit{Son})$. Si on lui présente la situation 6 pour la première fois, il répondra aléatoirement, car tant qu'il ne l'a pas déjà expérimentée, il n'a aucune information sur la façon dont la catégorie relative à la situation 6 peut ou pas lui procurer une récompense. Cette approche pose deux problèmes : l'agent n'est pas capable de généraliser et n'a aucune possibilité de s'améliorer au cours de son expérience, il faut créer un nouvel index pour toutes les nouvelles situations rencontrées. Cette solution n'est donc pas satisfaisante.

[1]Nous appelons réponse de l'agent le comportement provoqué par une situation de l'environnement, ici le fait de produire un son ou pas

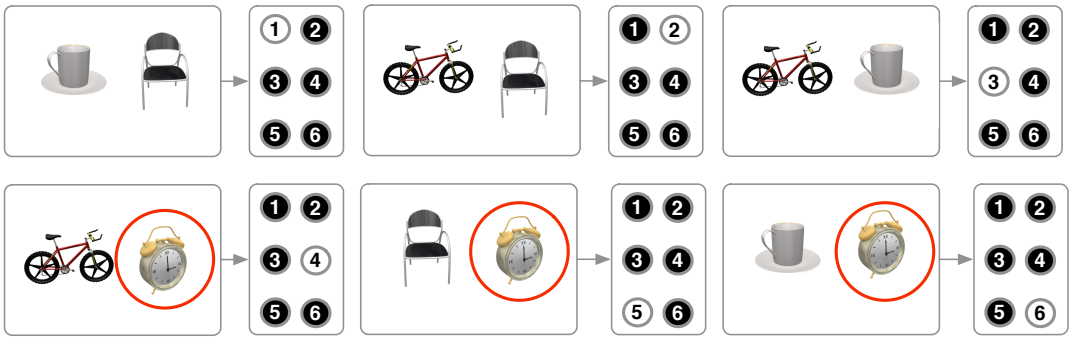


FIG. II.3 – Une unique catégorie par situation.

Pour que l'agent dispose de capacité de généralisation dans ce cas précis, il est nécessaire que son système de représentation indexe les formes présentées à gauche et à droite et pas les scènes dans leur globalité. Il y a donc plusieurs catégories à extraire simultanément dans chaque scène. Nous commençons à entrevoir que le système doit extraire simultanément plusieurs catégories. La figure II.4 illustre ce type de représentations.

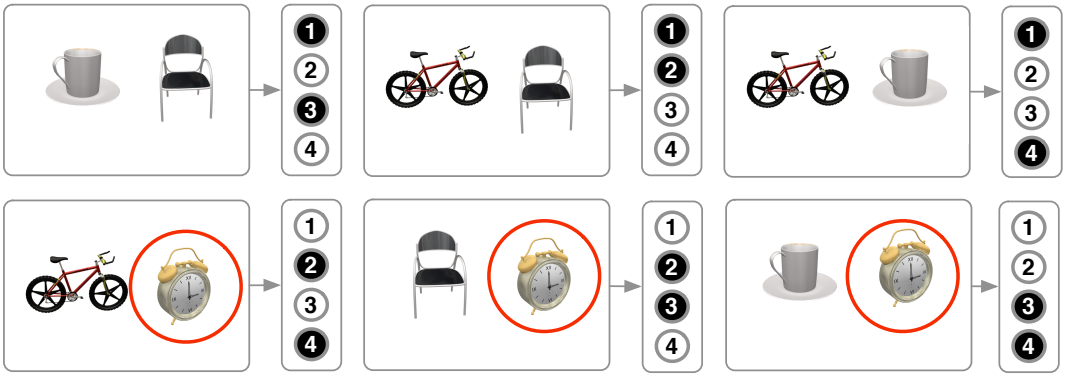


FIG. II.4 – Plusieurs catégories extraites simultanément par situation.

Dans ce cas, l'agent possède suffisamment d'informations pour généraliser les situations rencontrées, c'est à dire connecter l'action "produire un son" quand la catégorie 1 (le réveil) est active. Si une situation inconnue se présente et qu'elle contient au moins un réveil, alors la catégorie 1 s'activera quel que soit l'autre objet présenté, l'agent produira un son et sera récompensé. Cette approche est plus adaptée que la précédente mais possède encore des inconvénients, en particulier le fait que le nombre de représentations soit croissant avec le nombre de catégories relatives aux formes.

Si l'agent peut décomposer chaque situation en plusieurs catégories simultanément, il aura alors extrait de l'information nécessaire à la propriété de généralisation. Si l'agent possède des capacités de traitement suffisantes, il pourra utiliser cette information pour agir.

Cette expérience est très simple mais permet de mettre en évidence les capacités minimales qu'on attend de notre agent. En cela il doit au minimum réussir ce test, c'est à dire faire correspondre simultanément plusieurs catégories à chaque situation. C'est ce que nous appellerons représentations distribuées. Cette condition est nécessaire mais elle n'est cependant pas suffisante : l'agent devra posséder d'autres propriétés. Chaque fois que nous ajouterons une condition nécessaire, nous devrons faire en sorte qu'elle ne puisse pas rentrer en conflit avec les propriétés nécessaires déjà exhibées.

II.4.3 Représentations distribuées généralisées

Généralisons la propriété de représentation distribuée. Pour cela nous considérons le même dispositif expérimental que celui de la figure II.2. Cette fois nous supposons que la propriété cible n'est plus la forme mais le fait qu'une forme soit présente à une position particulière de l'espace. Nous ne présentons cette fois qu'un seul objet à la fois et nous faisons varier sa position. Il est présenté soit à gauche, soit à droite.

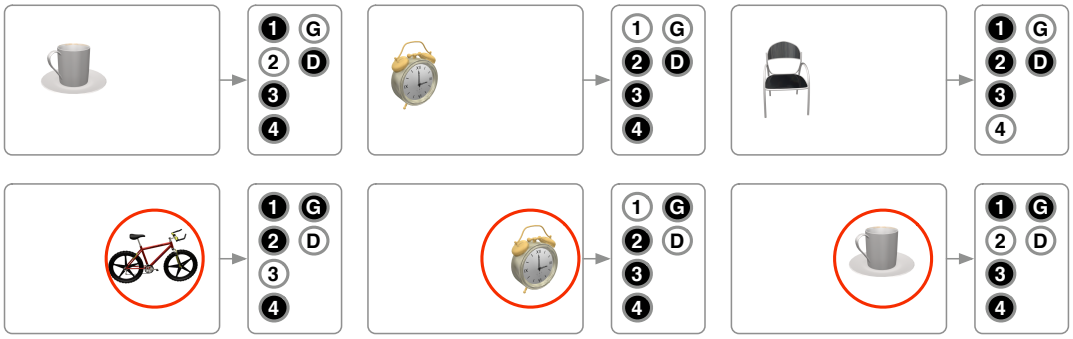


FIG. II.5 – Catégories de lieux et de formes.

La figure II.5 montre comment catégoriser à la fois la forme et le lieu. S'il n'y a pas une catégorie de lieu qui est extraite simultanément avec les catégories de formes, l'information nécessaire à la généralisation de situations n'est pas disponible pour l'agent. C'est le fait qu'une catégorie qui représente le lieu soit active systématiquement quand le système obtient une récompense en produisant un son qui est la cause de sa récompense.

L'objectif de l'agent est d'accroître son plaisir. Il doit donc trouver la cause de cet accroissement dans l'information qu'il extrait du monde. Nous avons vu que pour posséder des propriétés de généralisation, il était nécessaire de caractériser les situations en les décomposant à travers plusieurs catégories simultanément. Plus cette décomposition sera riche, plus l'agent

aura accès aux informations relatives aux causes de ses récompenses. Nous avons introduit une représentation distribuée basée sur la forme et la position. Nous pouvons la généraliser à tous les autres types d'informations perceptives, en particulier la couleur, la texture, la distance, ou encore le son. Le système doit réaliser une sorte de décomposition spectrale en composantes le plus indépendantes possibles de l'information présente sur ses capteurs.

L'environnement de l'agent dans cette expérience est extrêmement simplifié. Le monde ne contient que 4 formes et 2 lieux. Nous faisons l'hypothèse qu'il est parvenu à former des catégories à partir des données brutes de ses capteurs, jusqu'à extraire les notions de formes et de lieux. Ce niveau de complexité de la représentation de l'environnement est suffisant pour cette tâche particulière. L'agent n'a aucun besoin d'extraire autre chose que des objets ou des formes pour accroître le nombre de récompenses qu'il reçoit.

Supposons que la tâche consiste à produire un son pour une propriété géométrique particulière de la forme et non plus la forme elle-même. Par exemple, le fait d'avoir des pieds comme le réveil ou la chaise. Pour être en mesure de généraliser l'apprentissage de cette tâche à d'autres objets à pieds qu'il n'a jamais vus, par exemple une table, le système doit pouvoir représenter la catégorie pieds indépendamment de la forme à laquelle ce pied appartient.

Nous voyons que le statut de catégorie particulière que nous avons conféré aux formes globales que sont le réveil, le vélo, la tasse et la chaise est trop simple et est totalement dépendant de la tâche que nous avions initialement choisie au début de l'analyse. Ces catégories fondées sur les formes globales dans le cadre de cette nouvelle tâche sont alors aussi inadaptées que celles fondées sur les images globales dans l'expérience présentée figure II.3. De plus, associer une catégorie à chaque forme conduit également à une explosion combinatoire du nombre de catégories nécessaires pour représenter un objet sous toutes ses formes. Nous souhaitons réaliser un agent qui développe de nouvelles fonctionnalités. Au cours de son évolution, il va devoir réaliser des tâches qui étaient initialement inconnues. Sa capacité de représentation de l'environnement ne doit pas être liée à la réalisation d'une tâche particulière. Elle doit être compatible avec toutes les fonctionnalités dont l'agent dispose et néanmoins rester compatible avec toutes celles qu'il est susceptible d'acquérir.

Nous pouvons encore généraliser le caractère distribué des représentations en catégories élémentaires. La limite de cette décomposition se trouve dans les informations fournies par les capteurs. En d'autres termes, dans ce cas particulier, la fonctionnalité minimale de notre agent doit être d'associer la plus petite perception que fournissent ses capteurs à la production d'un son. Dans le cas général, la fonctionnalité sera d'associer la plus petite catégorie perceptive à l'action la plus élémentaire. S'il ne possède pas cette capacité de représentation, il n'aura pas à sa disposition l'information nécessaire à la généralisation des situations. Il n'aura donc pas suffisamment de souplesse pour apprendre des fonctionnalités plus complexes.

II.4.4 Représentations séquentielles

Nous pouvons étendre l'expérience initiale en proposant à l'agent de trouver des séquences d'événements dans un ordre précis. La figure II.6 montre un exemple de séquence qui peut être associée à une récompense. Le fait que l'agent doive pouvoir apprendre à répondre correcte-

FIG. II.6 – Séquence cible.

ment lorsque cette séquence lui est présentée nécessite que les représentations durent au moins aussi longtemps que la séquence qu'il doit reconnaître.

II.4.5 Unité de traitement : Extraction d'invariants perceptifs

Nous disposons maintenant de davantage d'éléments sur la façon dont l'information doit être représentée et traitée par le système. L'agent doit posséder un grand nombre d'unités de traitement ou représentationnelles travaillant en parallèle et dont l'activité doit refléter la présence d'une propriété élémentaire de l'environnement. Chaque situation sera décomposée par le système perceptif de la même manière qu'un prisme décompose la lumière blanche. Cette décomposition doit extraire des catégories relatives aux causes des variations observées par le système de représentations subjectives. Chaque unité aura donc un état qu'elle devra au moins communiquer au système d'action. Les catégories qui vont être extraites par ces unités de traitement ne sont pas connues à l'avance et devront être apprises. Pour cela le système devra extraire des invariants perceptifs. La figure II.7 montre les différentes unités de traitement qui composent le système de représentations objectives.

Ces unités de traitement s'activent de façon causale. Nous dirons qu'une unité extrait une propriété si elle s'active de façon causale à l'observation de cette propriété. Leur activité est relative au contenu des messages qu'elles observent. Chacune de ces unités de traitement doit extraire des catégories caractérisant des invariants perceptifs. Pour cela l'activité des unités de traitement doit posséder au moins deux propriétés nécessaires. La première est que pour deux observations similaires l'unité réponde de façon similaire. Nous appellerons cette propriété la **fidélité**. La seconde propriété est que si la scène que l'agent observe change très légèrement, alors la mesure proposée par l'unité doit également changer très peu. Nous appellerons cette propriété la **continuité**. Il y a deux phases dans le fonctionnement de ces unités. Initialement, elles ne sont pas adaptées à une propriété stable des données qu'elles observent et vont devoir se spécialiser. Il y a donc une période transitoire pendant laquelle la propriété qu'elles vont extraire n'est pas encore déterminée.

Si les mesures proposées ne possèdent pas ces deux propriétés, le système de représentation ne sera pas suffisamment stable pour que l'agent développe des comportements adaptés et il ne serait pas possible d'associer une perception à une action[2].

Nous appellerons **persistance** la durée pendant laquelle les unités de traitements restent actives. Il peut arriver que les situations que l'agent observe soient obstruées par des objets

[2]Un système de représentation non-stable entraîne une impossibilité d'associer des représentations à des actions

pendant une courte durée. Pour autant l'agent doit conserver une représentation stable de l'environnement. Lorsque la cause de l'activation a disparu, il ne faut pas que cette information soit perdue par l'agent. Si c'était le cas, tous les comportements engagés à cet instant se trouveraient détachés de la cause de leur déclenchement. La propriété de persistance de l'activité est suffisante pour éviter ce phénomène de perte de représentations.

II.4.6 Compétition

Le système ne dispose que des données brutes fournies par ses capteurs. Chaque capteur observe une propriété distale de l'environnement à travers sa projection proximale. Ces mesures ont à la fois des propriétés spatiales et temporelles. Chaque capteur mesure une propriété particulière de l'environnement dans une région de l'espace donnée et propose cette mesure pendant une durée donnée. Ceci est une précision de nos hypothèses relatives aux entrées et aux sorties de l'agent. Nous appellerons champs récepteur[3] l'ensemble des unités de traitement observées par une unité de traitement donnée. Nous appellerons espace conjugué la région de l'espace observée par une unité de traitement. La figure II.7 montre un exemple de champs récepteurs et d'espace conjugué.

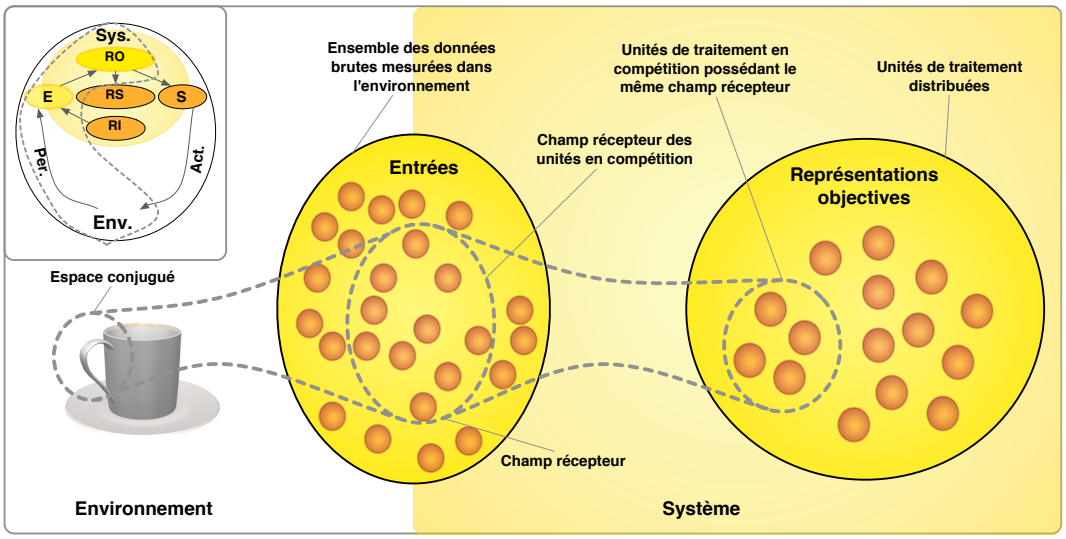


FIG. II.7 – Compétition pour l'extraction d'une propriété perceptive.

D'une part, les unités de traitement qui composent le système de représentation du système doivent chacune apprendre à extraire une catégorie. Initialement, les unités de traitement ne sont pas spécialisées. D'autre part chaque région de l'espace peut être remplie par un ensemble

[3]Le terme champ récepteur est emprunté aux neurosciences, nous lui donnons ici un sens similaire

de propriétés différentes. Il doit donc y avoir plusieurs unités de traitement pour chaque région de l'espace. Il ne faut pas que toutes les unités relatives à une région de l'espace se spécialisent dans l'extraction de la même information, elles vont devoir se différencier. Si elles apprenaient à extraire la même propriété, l'agent perdrait inutilement des ressources représentationnelles et ne serait plus en mesure d'extraire certaines informations qui pourraient s'avérer importantes durant son évolution. L'adaptation d'une unité de traitement à une propriété devra donc se faire à travers une compétition.

Le fait que les unités de traitement soient en compétition pour se déterminer implique que leurs activités soient comparables. C'est à travers un ordre dans les réponses de ces unités qu'il sera possible de désigner un gagnant et des perdants. En plus de la fidélité et de la continuité, nous ajoutons donc comme propriété la **comparabilité**.

II.4.7 Représentations hiérarchiques

La figure II.7 montre un exemple d'unités de traitement en compétition observant les mêmes données brutes. Considérons la question de la taille des champs récepteurs des unités de traitement du système de représentation objective. Nous avons vu que l'hypothèse naïve initiale qui consistait à associer chaque situation à une catégorie ne permettait pas d'extraire les informations nécessaires pour que l'agent généralise les situations nouvelles qu'il n'a jamais vues. Les champs récepteurs des unités de traitement ne doivent donc pas couvrir la totalité des données brutes, mais seulement des parties de l'ensemble des données brutes.

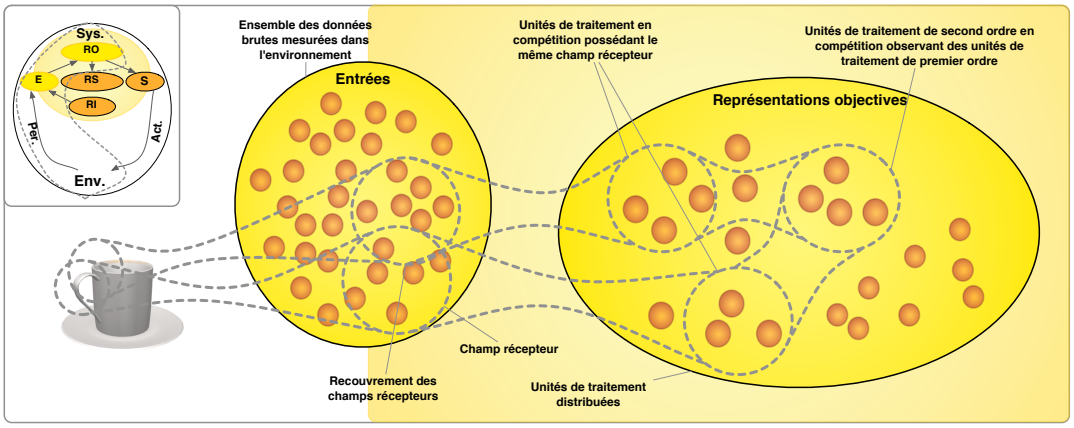


FIG. II.8 – Représentations hiérarchiques.

Pour que les unités de traitement aient une chance d'extraire des propriétés stables de ces données, les champs récepteurs doivent couvrir des parties connexes. Si les données observées sont voisines dans l'espace, alors leurs lois de variation seront en général liées, c'est à dire qu'elles véhiculeront une information mutuelle non nulle.

Le nombre d'états observables à travers un ensemble de données brutes augmente exponentiellement avec le nombre d'unités. Prenons un exemple simple : supposons que les données brutes soient binaires $\{0,1\}$. Ce cas constitue le codage le plus simple. Le fait de considérer un champ récepteur contenant n données brutes permet de coder 2^n états possibles. Si les unités de traitement observent un trop grand nombre de données brutes, le nombre d'unités en compétition dans le système de représentation objective sera sujet à une explosion combinatoire. Les unités de traitement doivent se spécialiser dans l'extraction d'invariants. Elles doivent aussi former une base des contenus perceptifs proposés par les données brutes qu'elles observent.

Certaines unités de traitement doivent pouvoir rendre compte de situations globales, c'est à dire de l'état de l'ensemble des données brutes. Si des unités de traitement observent des parties connexes de l'ensemble des données brutes, alors il doit exister dans le système de représentation des unités de traitement qui observent ces unités là pour rendre compte de la totalité des données brutes. C'est ce que nous appellerons les représentations hiérarchiques (voir figure II.8). Les unités de traitement sont donc organisées en graphes.

Plus on progresse dans le graphe en s'éloignant des données brutes, plus l'activité des unités de traitement doit expliciter des causes stables des variations d'états subjectifs. En explicitant ces causes, elles simplifient le processus de généralisation nécessaire à l'adaptation des actions de l'agent.

II.4.8 Sémantique et activation des unités de traitement : codage par position

Le système représente l'information de façon totalement distribuée. Les éléments de base de cette décomposition sont les unités de traitement. L'hypothèse la plus simple consiste à considérer que toutes ces unités sont formellement équivalentes. Posons nous la question de la sémantique véhiculée par l'activation de ces unités. Contrairement aux systèmes symboliques où un sens est attribué aux symboles manipulés et, où il faut créer autant de symboles que d'objets dans le monde, cette façon de représenter l'information est ouverte. Il n'est pas nécessaire que le concepteur du système attribue un sens aux unités de traitement. C'est leur position dans le graphe, ou le réseau, qui va donner un sens au fait qu'elle s'active ou non. Si une unité de traitement s'active à proximité d'une entrée visuelle, l'information qu'elle véhiculera sera relative à une petite région de l'espace et traduira la présence d'une propriété géométrique élémentaire dans l'espace conjugué. A titre d'exemple, si une unité de traitement observe des images produites par une caméra, son activation pourra être relative à une orientation locale[4] présente dans une petite région de l'espace. Une unité de traitement placée plus loin dans le système pourra observer une région plus grande et extraire une information relative à la présence d'une forme plus complexe, par exemple un coin. Ce que produira ce système de représentation est indépendant du concepteur. Il s'agit d'un codage par position. C'est le même principe qui a été utilisé dans le système de numération par position.

[4]Cette orientation locale peut être un bord ou un trait orienté

Ouvrons une parenthèse historique sur cette découverte mathématique fondamentale qui peut encore aujourd'hui être considérée comme un monument de l'esprit humain. Les premiers systèmes de numération étaient limités. Ils utilisaient un ensemble de symboles pour désigner des quantités. Ce type d'approche présente deux problèmes majeurs : pour exprimer de grands nombres il est nécessaire de créer de nouveaux symboles, et surtout il est impossible de calculer de façon systématique. Si on ne connait pas un symbole, on n'a aucun moyen de le comprendre. L'invention des systèmes de numération par position a réglé ce problème de façon très élégante. Ce système nécessite l'existence d'un signe qui signifie le vide, l'absence d'une décimale. Il y a des traces de ce type de numération dans le système de numération babylonien au IIIe siècle avant notre ère, et dans le système maya au cours du premier millénaire de notre ère. La notion de zéro complet est le plus souvent attribuée à un texte sanskrit datant du cinquième siècle de notre ère. On considère également qu'Al - Khuwarizmi $(783 - 850)$ a introduit les premiers calculs arithmétiques en utilisant ce système de numération. Les indiens ont représenté le zéro comme un cercle. Ce symbole était nommé "Sunya" qui signifie "vide" en langue indienne (sanskrit). Traduit en arabe, Sunya, devient "Sifr" (vide). L'avantage immense de ce système de numération est qu'il permet d'exprimer tous les nombres possibles. Il n'est plus nécessaire de créer de nouveaux chiffres pour exprimer de très grands nombres : Il est ainsi possible de tous les comprendre. L'infinité des quantités est rejetée dans l'infinité des positions.

Revenons au système de représentation de notre agent. Nous souhaitons qu'il dispose d'un système de représentation ouvert, c'est à dire qui puisse permettre de créer de nouvelles représentations sans nécessiter la création de nouveaux symboles. Il doit être en mesure de représenter sans l'intervention du concepteur, des notions qu'il ne connait pas.

Utiliser un système de codage par position permet d'inclure la notion de sémantique dans la position. Nous avons vu que la propriété nécessaire de compétition impose que les niveaux d'activité des unités de traitement soient comparables. Nous pouvons supposer que toutes les unités de traitement sont formellement identiques. Si une unité est inactive, son inactivité reste cependant observable par les autres unités. Cette inactivité constitue le vide de ce codage par position. Ainsi la logique du système n'est pas uniquement positive, la description du monde n'est pas l'ensemble des choses vraies à chaque instant, mais l'ensemble de choses vraies et fausses. C'est le lieu et l'état de l'unité de traitement qui donne son sens. S'il est nécessaire pour l'agent de devoir percevoir une nouvelle propriété de l'environnement, une assemblée d'unités de traitement pourra se spécialiser dans l'extraction de cette propriété sans qu'il soit nécessaire de créer un symbole particulier. Ce système est ouvert parce qu'il permet d'exprimer des notions qui ne sont pas initialement connues. Ce point précis constitue le fondement de cette thèse.

II.4.9 Représentation des actions propres : Boucle sensori-motrice

Jusqu'à présent nous n'avons pas parlé des actions du système. Nous avons fait l'hypothèse que le système possède des effecteurs. Certaines unités doivent être directement connectées à ces effecteurs. Elles réalisent un traitement inverse de celui des entrées. L'activité de ces unités a la propriété de modifier localement l'environnement.

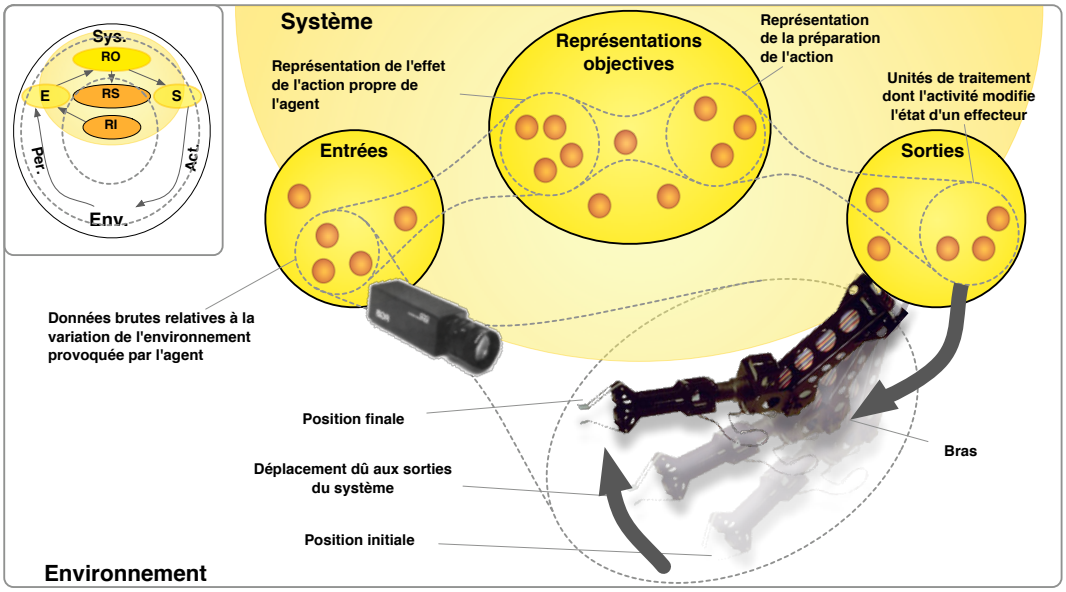


FIG. II.9 – L'agent doit pouvoir observer l'effet de ses propres actions.

L'agent doit pouvoir apprendre de façon autonome. Pour cela il doit expérimenter son environnement et être en mesure de percevoir les modifications de l'environnement qu'il a lui-même occasionnées ainsi que l'effet subjectif que le résultat de ces modifications produit sur lui. De cette façon, il sera en mesure de créer de nouvelles représentations relatives à l'effet de ses propres actions sur le monde. La figure II.12 illustre cette propriété. L'agent possède un bras manipulateur dont il ne sait initialement rien. Il doit être en mesure de déplacer son bras et d'extraire deux types d'informations : le fait qu'il a agi, c'est à dire que des unités de traitement directement connectées aux sorties du système se sont activées un peu avant que le bras bouge et qu'il ait perçut une variation du monde à l'aide de ses capteurs visuels. Il a suffisamment d'informations dans ce cas pour pouvoir statistiquement extraire la cause de cette variation du monde et la réutiliser.

Le système ne connaît initialement pas ses effecteurs et doit apprendre à les utiliser en explorant les différentes façons de les actionner. Toutes les paires de capteurs et d'effecteurs ne sont pas équivalentes en termes d'efficacité. Dans tous les cas, l'environnement joue le rôle de retour. Nous pouvons donner quelques exemples de couples d'effecteurs et de capteurs qui possèdent cette propriété : Un haut parleur et un micro, un bras manipulateur et une caméra, etc. Dans le cas d'un haut parleur et d'un micro, le retour se fait sous la forme d'ondes sonores, dans le cas du bras et de la caméra le retour se fait à l'aide des propriétés optiques de l'environnement, comme l'éclairage de la scène, la couleur du bras, le fait qu'un objet empêche

que la caméra observe le bras. Dans notre exemple initial, le retour se fait par le tuteur : c'est lorsque ce dernier juge que le système à répondu de façon correcte que l'agent peut percevoir un changement d'état sur le bouton de récompense ou de punition.

D'une manière générale, l'agent expérimente son environnement en agissant pour essayer de réactiver les représentations subjectives positives qu'il possède à travers des variations du monde qu'il provoque.

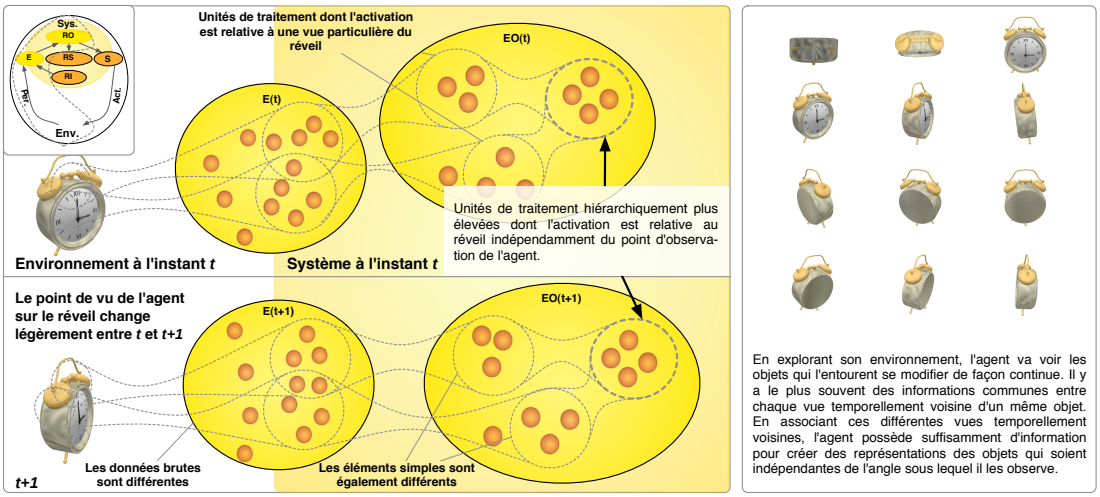
Cette boucle sensori-motrice impose une contrainte temporelle. Lorsque l'information pénètre dans le système, elle se propage dans les réseaux hiérarchiques d'unité de traitement. Chaque couche nécessite un certain temps pour être traversée. Plus il y a de couches, plus le traitement des données arrivera tard et plus l'agent prendra de temps pour produire une action appropriée à une situation perçue. Le temps minimum de traversée du système de représentation caractérise la dynamique de l'agent. Il ne pourra être adapté qu'à des phénomènes compatibles avec son temps de réaction minimal.

II.5 Stabilisation des représentations : Associations et causalité

Nous avons vu que l'agent devait s'adapter en réalisant des associations entre unités de traitement en se basant sur la notion de causalité. La capacité de représentations stables est centrale. L'agent doit pouvoir trouver des propriétés connues et stables dans des situations inconnues et nouvelles. Pour cela, il doit être en mesure de construire des catégories toujours plus découplées des données brutes. La capacité à associer des événements joue un rôle important. Il doit associer des composantes perceptives indépendantes et des composantes actives indépendantes pour créer de nouvelles fonctionnalités. Ces nouvelles fonctions comportementales devront être réutilisables dans des schémas plus complexes. Le moteur de l'apprentissage est basé sur la capacité d'associer des événements causaux. Nous montrons ici que les propriétés de stabilisation des représentations perceptives sont dans un premier temps généralisables aux propriétés de l'environnement pour une même modalité perceptive, et que cette propriété peut s'étendre à des associations multi-modales.

II.5.1 Associations temporelles et spatiales : l'indépendance au point de vue

L'objectif de l'agent consiste à accroître l'effet subjectif qu'il perçoit. Pour cela il doit trouver les causes de ses variations d'effets subjectifs dans les représentations des situations qu'il rencontre pour faire en sorte que les événements à l'origine de ces causes, c'est-à-dire ses actions, soient à nouveau produites dans des situations similaires. Si la cause de cet accroissement est liée à la présence d'un objet ayant une propriété particulière, l'agent maximisera ses chances de succès s'il est capable de retrouver cet objet quel que soit l'angle sous lequel il l'observe. Il est donc nécessaire qu'il dispose d'unités de traitement qui se spécialisent dans



En explorant son environnement, l'agent va voir les objets qui l'entourent se modifier de façon continue. Il y a le plus souvent de l'information commune entre chaque vue temporellement voisine d'un même objet. En associant ces différentes vues temporellement voisines, l'agent possède suffisement d'information pour créer des représentations des objets qui soient indépendantes de l'angle sous lequel il les observe.

FIG. II.10 – Association temporelle de forme, indépendance à la vue.

la perception d'objets de l'environnement indépendamment de l'angle sous lequel l'agent les observe.

L'agent est situé dans le monde, ainsi il l'observe sous un angle particulier. En tant que partie du monde lui-même, il ne peut accéder aux propriétés de l'environnement qu'à travers des projections de celui-ci sur ses capteurs. Il ne peut accéder visuellement aux stimuli distaux qu'à travers leurs images proximales et ne peut disposer que d'une fraction de la réalité en observant le monde. Pourtant il doit généraliser ces différentes perceptions, c'est à dire que s'il observe un objet sous un angle différent, il doit à la fois savoir que c'est le même et qu'il a changé de point de vue par rapport à lui.

Si des unités de traitement sont supposées s'activer pour un même objet, quel que soit le point de vue sous lequel il est observé, alors il est nécessaire qu'elles se spécialisent dans l'association temporelle de ces différentes vues. Pour cela la propriété de persistance est nécessaire. La figure II.10 illustre cette propriété en prenant comme exemple une assemblée d'unités de traitement codant un réveil, quel que soit l'angle sous lequel il est observé. Lorsque l'agent expérimente lentement son environnement, les vues successives sous lesquelles il observe son environnement varient peu. Les vues successives qu'il observe contiennent de l'information

mutuelle. Outre le voisinage temporel, il est très probable de constater très peu de changement sur au moins une propriété de l'objet parmi sa position, sa taille, sa couleur ou sa texture, au moins à une échelle spatiale particulière. La propriété de continuité assure que durant cette variation, les unités de traitement distribuées continuent à s'activer. Il est donc très probable qu'il existe un sous-ensemble d'unités qui restent actives entre deux vues successives. Ces unités pivots doivent jouer un rôle dans l'apprentissage d'associations temporelles.

Les unités de traitement qui composent le système de représentation objectif sont organisées en graphe. Le fait que le système de représentation objective soit supposé fermer une boucle sensori-motrice assure qu'il existe des chaînes d'unités de traitement qui conduisent des entrées aux sorties du système. Les unités proches des entrées doivent coder les propriétés objectives simples de l'environnement. Les unités proches des sorties doivent coder les composantes des différents degrés de liberté. Les informations dont nous disposons sur la façon dont l'agent organise ses représentations laisse supposer que les unités codant une invariance aux points de vues, pour une propriété géométrique du monde, sont à la fois éloignées des entrées et des sorties.

Pour mieux généraliser, l'agent doit être en mesure de trouver des propriétés stables quand le mode change. Plus une unité de traitement est éloignée des entrées et des sorties plus elle doit stabiliser l'information qu'elle véhicule, c'est-à-dire que son activité doit persister alors que les causes de son activation varient. On peut en conclure que ces unités doivent être en mesure de s'activer pour des catégories complexes caractérisant des propriétés de l'environnement indépendamment du point de vue de l'agent et plus seulement pour des propriétés géométriques simples. Plus on s'éloigne des entrées, plus les catégories extraites par les unités de traitement se découplent de données brutes particulières qui ont permis leur activation.

II.5.2 Vers des représentations indépendantes des modalités perceptives

De la même façon que l'agent doit posséder des unités de traitement codant pour un objet particulier dans son système de représentation objective, il doit être en mesure de coder un objet indépendamment de sa modalité d'observation.

L'agent a accès au monde à travers plusieurs types de capteurs. Les objets qu'il observe ont probablement une composante sur chaque modalité (par exemple une image, un son, etc.). Si l'agent entend la signature d'un objet ou perçoit son image, une assemblée doit coder le fait qu'il s'agit de deux facettes de la même réalité. La figure II.11 illustre cette propriété. Les unités qui codent pour des classes de propriétés du monde indépendamment de la modalité qui a permis de les activer permettent à l'agent de représenter des objets de façon abstraite. Dans le cas d'un agent disposant de capteurs visuels et auditifs, ce type d'association permet en particulier à l'agent d'associer un nom prononcé par un expérimentateur et un objet quel que soit son point de vue. L'agent peut également associer la façon de se déplacer d'un objet avec un son prononcé par l'expérimentateur. Ce type d'association permet de construire les premiers éléments d'un système de désignation orale des objets (nom communs) et de leur façon de se mouvoir (verbes élémentaires). Ceci peut constituer les bases d'un langage.

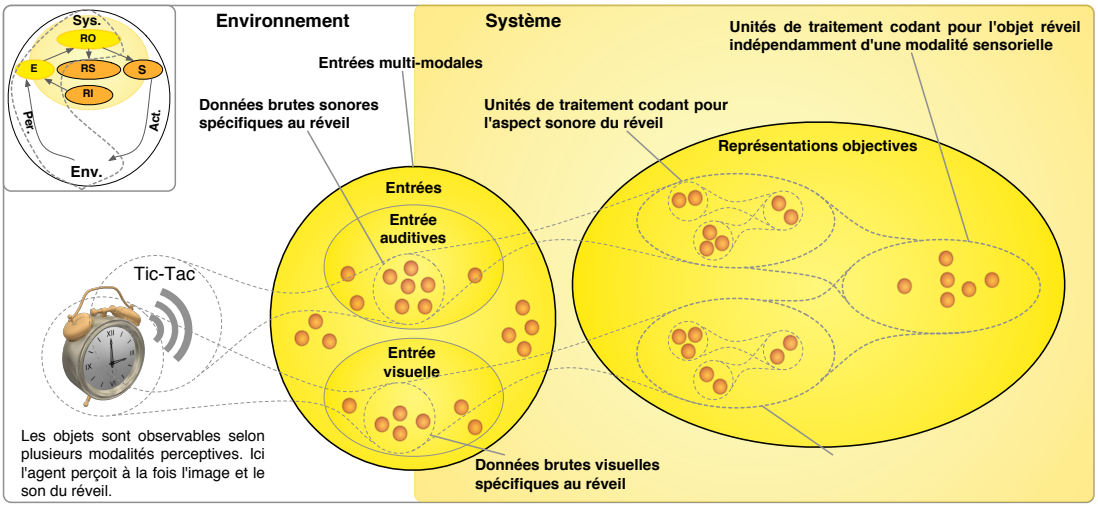


FIG. II.11 – Association temporelle de modalités perceptives, indépendance à la modalié.

II.6 Questions soulevées par un système de représentation totalement distribué

II.6.1 Liage temporel des assemblées d'unités de traitement

Le monde réel est complexe. L'agent sera très souvent amené à devoir agir en fonction de plusieurs objets présents simultanément. Le fait de représenter de façon distribuée l'ensemble des propriétés qui constituent les parties indépendantes de l'environnement pose un problème important illustré par la figure II.12. Si l'agent décompose le monde en un ensemble de propriétés indépendantes, comment peut-il retrouver l'unité des objets ? Dans l'exemple que nous prenons nous considérons que le système possède deux objets distincts dans son champ visuel : une tasse et un réveil. La tasse occupe le quart d'espace supérieur gauche et le réveil le quart d'espace inférieur droit. Supposons que le système sache que regarder la tasse produit un accroissement de l'état subjectif, c'est à dire que le fait de déplacer sa caméra de façon à ce que l'image de la tasse se retrouve au centre de la caméra produise un accroissement de l'état subjectif. Pour ne pas agir de façon aléatoire, il doit savoir où se trouve la tasse. Il doit exister dans le système un moyen de retrouver l'information de position de la tasse, et un lien entre l'assemblée dont l'activité est relative à la forme de la tasse et l'assemblée dont l'activité est relative à sa position. Or, le système de représentations objectives doit utiliser un codage par position ; la position des unités de traitement dans le réseau permet de coder le sens véhiculé par leur activation.

La solution la plus simple consiste à envisager l'existence d'unités de traitement qui observent l'ensemble des combinaisons de paires de propriétés. Dans ce cas particulier, il faut que ces unités observent toutes les combinaisons de formes et de lieux. Pour chaque lieu, il

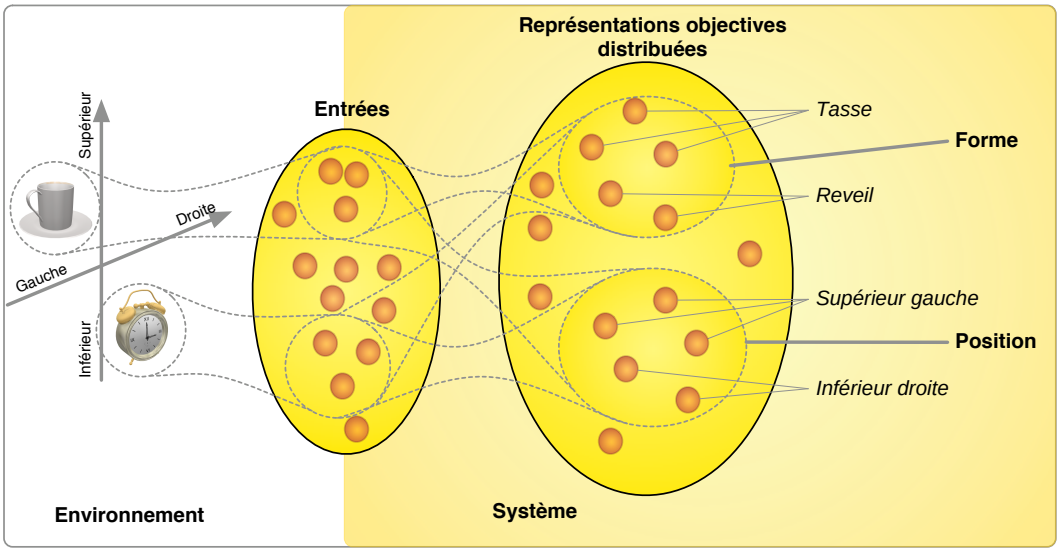


FIG. II.12 – Problème lié à l'unité des objets.

devra exister autant d'unités que de propriétés observables. Cela conduit à une explosion combinatoire du nombre d'unités relatives à ces paires de propriétés. Cette solution n'est donc pas envisageable.

Cet exemple se généralise à n'importe quelle paire de propriétés distribuées caractérisant une partie du monde et connaissant l'intérêt de l'une, le système aurait besoin de l'autre information pour agir.

Supposons que l'agent évolue dans un monde simplifié à l'extrême ne contenant qu'un seul objet. Toutes les représentations actives dans le système à la suite de la perception de cet objet seront relatives à lui et seulement à lui. Il n'y aucune ambiguïté dans ce cas, il n'est pas possible d'attribuer une fausse position à une forme par exemple. Si l'agent a besoin de la position d'un objet, il n'y aura qu'une assemblée relative à cette position active dans le système à ce moment là. Toutes les représentations activent à ce moment là sont liées par le fait qu'elle représentent cet objet. La figure II.13 montre la décomposition spécifique de la tasse et du réveil dans ce cas.

D'une manière générale, si le système ne s'intéresse qu'à une propriété du monde à la fois, alors toutes les représentations objectives qu'il sera capable d'extraire de ses entrées seront relatives à cette propriété. Si l'agent s'intéresse à plusieurs propriétés, il serait intéressant qu'il puisse superposer plusieurs représentations simultanément.

Il est possible d'utiliser le temps pour réaliser ce liage en évitant une explosion combinatoire. Dans ce cas il faut considérer l'activité des unités de traitement comme des fonctions périodiques, et faire en sorte que l'assemblée correspondant à la tasse et l'assemblée corres-

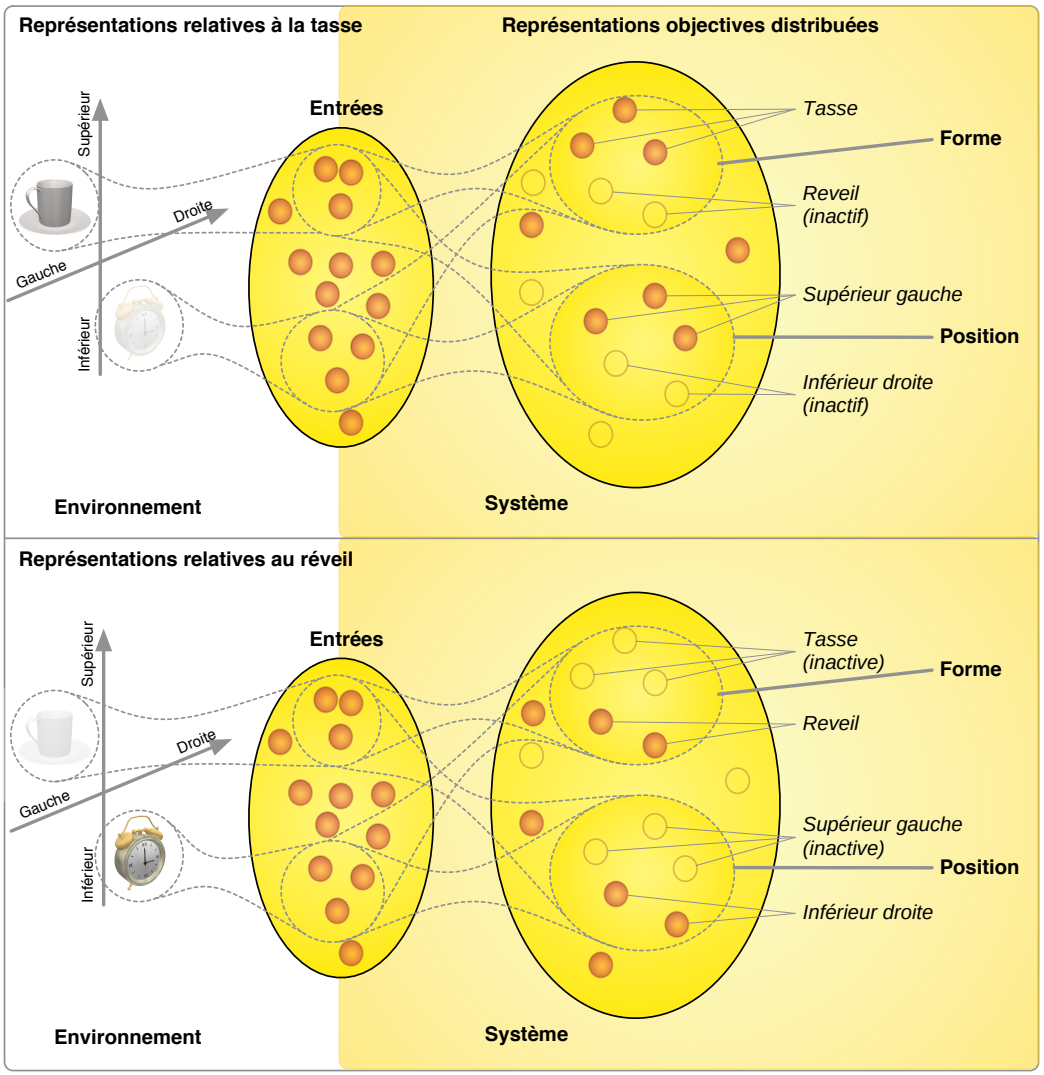


FIG. II.13 – Superposition de plusieurs états.

pondant à la région supérieure gauche soit en phase, et que l'assemblée correspondant au réveil et celle correspondant à la région inférieure droite soit également en phase, mais sur une phase différente.

L'unité des parties de l'environnement doit être représentée par le système comme des assemblées d'unités de traitement s'activant périodiquement sur une phase distincte. Chaque assemblée doit posséder sa propre phase et coder pour une propriété particulière de l'environnement.

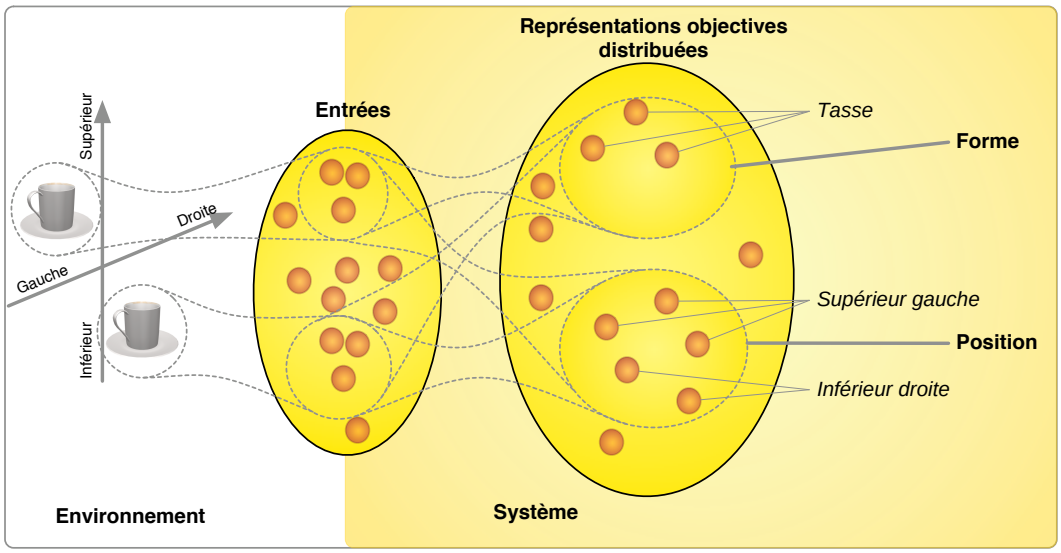


FIG. II.14 – Problème lié à l'observation de plusieurs objets identiques.

II.6.2 Superposabilité des états d'activation

Revenons sur l'exemple précédent et remplaçons le réveil par une tasse de façon à avoir deux objets identiques mais à des positions différentes. Cette fois, la représentation distribuée de la tasse devra être liée à deux positions, c'est-à-dire qu'elle doit être en mesure de superposer ses états. Si elle n'est pas en mesure de superposer deux états simultanément alors la phase des deux lieux serait la même et ces deux propriétés indépendantes dans l'environnement seraient perçues comme une unité par le système.

La fonction périodique engendrée par la superposition de ces deux fonctions devra contenir l'information relative à l'activité des deux propriétés extraites. Les unités de traitement devront être en mesure d'observer indépendamment chacune de ces propriétés. En particulier une unité de traitement devra pouvoir extraire la phase et la valeur d'activation des unités de traitement qu'elle observe.

Si on considère que chaque unité de traitement possède un procédé unique pour extraire de l'information qu'elle observe, alors chaque fonction périodique devra avoir la même forme.

La propriété de superposition impose une contrainte sur le nombre de signaux superposables qu'une même unité peut traiter durant chaque période. Ce nombre dépend à la fois de la forme du signal et de la capacité de l'unité à séparer deux signaux. Intuitivement, un signal impulsionnel répond au moins à la seconde exigence en permettant de rendre chaque composante totalement indépendante et simplifie considérablement leur séparation.

II.7 Construction des unités perceptives : Mise en synchronie des assemblées

L'environnement est composé d'un grand nombre d'objets. Il est très rare que l'agent soit confronté à un objet unique. Le cas idéal où toutes les représentations actives à un instant donné ne codent qu'un seul objet est lui-même très rare. Rappelons que dans ce cas si l'agent possède dans son système de représentation objective une assemblée codant pour une région de l'espace indépendamment de son contenu et une assemblée codant pour une forme indépendamment de son lieu, alors il est sûr que l'assemblée correspond à une même propriété de l'environnement. Si l'agent s'intéresse à cette forme, il pourra la trouver dans la région de l'espace codée par l'assemblée de lieu.

Quel mécanisme perceptif peut conduire à ce que des parties de la réalité se superposent dans le système de représentation ? Comment l'agent peut créer ces groupes d'assemblées synchrones si le monde est complexe et contient plusieurs propriétés simultanément ?

II.7.1 Le rôle particulier des représentations de lieux

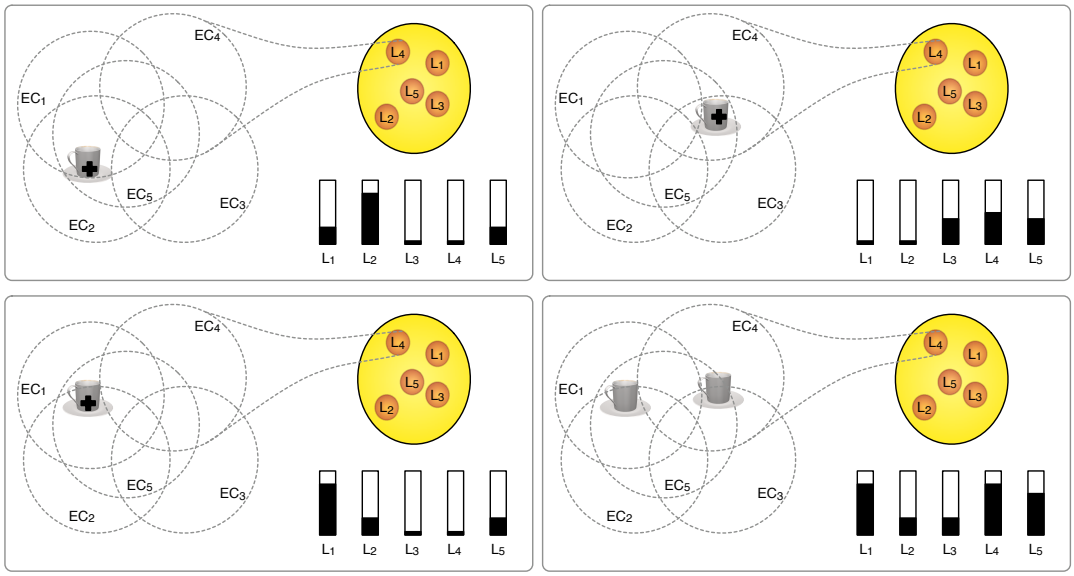


FIG. II.15 – Il n'est pas possible d'extraire plus d'un lieu à la fois en se basant sur la forme.

Le système que nous étudions observe son environnement de façon active. Il doit choisir ce qu'il observe. Son point de vue sur le monde dépend de l'endroit où il se trouve et de la direction d'observation de ses capteurs. En se déplaçant ou en dirigeant ses capteurs autrement, il va observer le monde sous un angle différent. Les représentations objectives de l'agent doivent

être totalement distribuées. De ce fait, les assemblées qui codent pour des lieux ne portent pas d'information sur les autres propriétés associées à ces lieux. Quel mécanismes sont nécessaires à l'extraction d'une information de lieu pure ?

La figure II.15 montre qu'au-delà de deux objets, il n'est plus possible d'extraire une information de lieu simplement. Dans ce cas particulier, le système possède 5 unités (L_i) de traitement codant pour des lieux dans son système de représentation objective. Chacune de ces unités de traitement est relative à un espace conjugué dans l'environnement. Si un objet est présent dans cet espace conjugué (EC_i), alors l'activité de l'unité est d'autant plus forte que l'objet observé est proche du centre. Si deux objets sont présents dans l'espace conjugué d'une unité, la réponse est la somme des réponses engendrées par chaque objet pris individuellement. Ce type de traitement de l'information est un cas particulier. D'une manière générale cet exemple permet de comprendre qu'à partir de deux objets la notion de lieu est ambiguë. Il n'est alors plus possible d'extraire une information de lieu simplement.

Il semble nécessaire que l'agent ne s'intéresse qu'à une propriété de l'environnement à la fois en « cachant » artificiellement les autres. Si le système sait réaliser ce choix, il pourra alors mettre en synchronie l'ensemble des unités relatives à la décomposition perceptive de la propriété sélectionnée. Comment réaliser ce choix ?

II.7.2 Représentations distribuées de lieux et de formes : Où ? Quoi ?

Nous avons vu que l'agent doit représenter l'environnement de façon totalement distribuée. Considérons le cas où l'agent dispose d'une caméra. Supposons qu'il dispose d'autant d'assemblées susceptibles de coder qu'une tasse est présente dans l'image que de positions possibles pour cette tasse (voir figure II.16). Ces assemblées coderaient alors à la fois une forme et un lieu.

L'agent à besoin d'information sur son environnement pour agir de façon appropriée. Une action possible peut être la préhension d'un objet qu'il perçoit dans son champ visuel[5]. Pour agir il a besoin d'avoir une information sur la nature de l'objet et l'endroit où il se trouve. Si ces deux informations sont véhiculées par la même assemblée, alors si l'agent apprend à approcher son bras du lieu où se trouve l'objet, il ne sera pas en mesure de généraliser l'apprentissage de cette notion de lieu à d'autres catégories objets. Avec un codage de forme et de position de ce type, l'agent aura autant de lieux que de catégories perceptives.

L'exemple considérant le cas de la préhension est généralisable à toute action utilisant une information de position. On comprend avec cet exemple que pour pouvoir généraliser l'apprentissage d'actions relatives à la position des objets quelle que soit la forme des objets, il est important que l'information de positions soit codée par des assemblées indépendantes.

Le système de représentation de lieux doit donc être indépendant du système de représentation de la nature des objets, dans le cas d'informations visuelles, la position doit être représentée indépendamment de la forme, de la couleur, et de la texture des objets.

[5]On appelle ici champ visuel la réunion des espaces conjugués des unités de traitement observant directement ou non les données brutes issues de la caméra

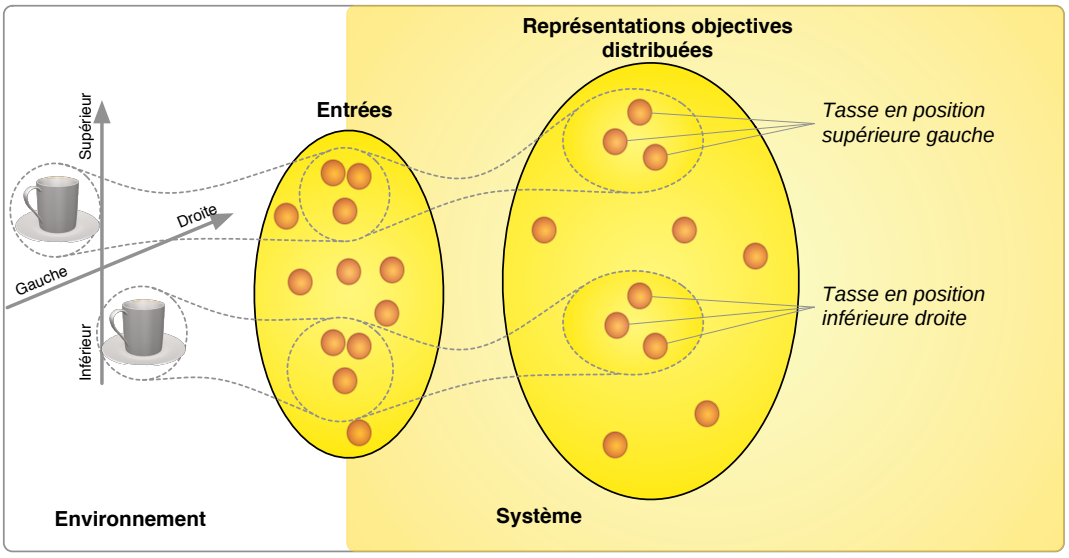


FIG. II.16 – Si des assemblées codent à la fois la position et la forme le système ne peut pas généraliser certaines actions.

De plus donner au système de représentation la capacité de catégoriser n'importe quelle forme à n'importe quelle position conduirait à une explosion combinatoire. Seule une position privilégiée dans l'image est suffisante pour que le système catégorise la forme qui s'y trouve. Un autre système est donc nécessaire pour déplacer la caméra de telle sorte que l'image des objets perçus se retrouve séquentiellement à cette position et que le système puisse les catégoriser.

II.7.3 Carte de saillance et régions d'intérêt

Choisir une propriété de l'environnement et cacher les autres sous-entend que cette propriété possède suffisamment d'intérêt pour le système. Or, l'agent ne peut s'intéresser à une propriété sans la connaître. Si l'agent veut observer une région de l'espace privilégiée c'est bien pour prendre connaissance des propriétés qu'elle contient en décomposant cette propriété dans le système de représentation objective. Si l'agent s'intéresse à une région particulière il doit le faire avant de savoir ce que cette région contient en terme de représentations objectives.

Supposons que la partie supérieure gauche du champ visuel de l'agent contienne une tasse. Si l'agent le sait déjà, il n'a aucun besoin de l'observer. Ce qui guide le fait qu'il choisit d'observer cette tasse est un autre type de représentation objective simple et qui ne nécessite pas d'apprentissage.

Nous avons vu qu'il est préférable de centraliser les unités de traitement relatives à l'extraction de contenu dans une région particulière. Pour des raisons de symétrie, cette position

particulière sera choisie au centre de l'image. Catégoriser l'information contenue dans une région de l'espace consiste donc à placer cette région de l'espace au centre de l'image. Pour cela le système doit déplacer sa caméra.

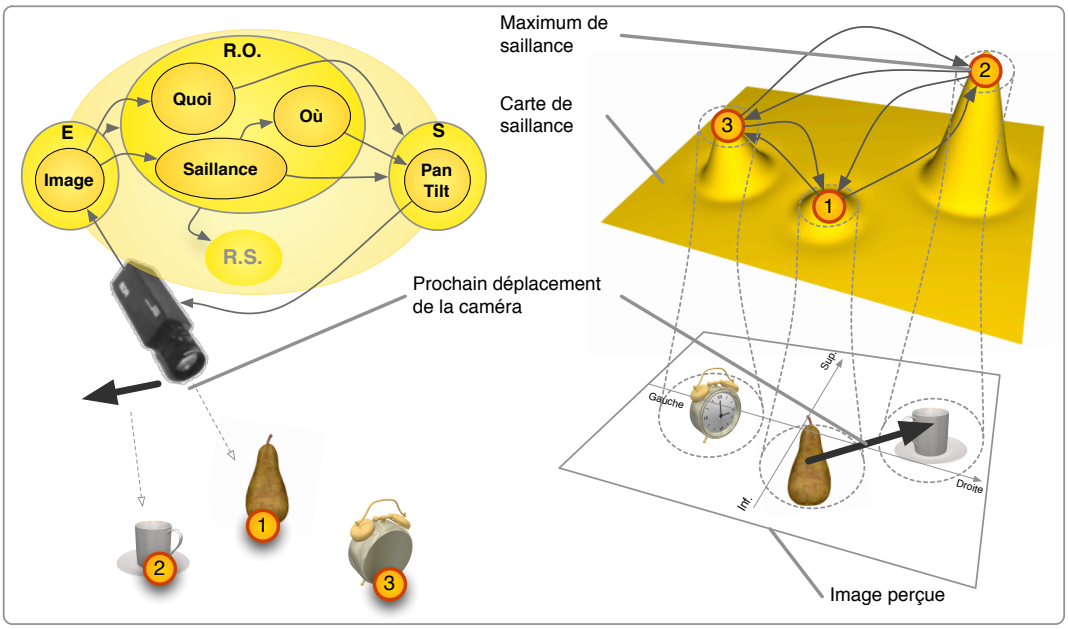


FIG. II.17 – La carte de saillance permet au système de choisir sa prochaine observation sans avoir à catégoriser l'objet suivant avant de l'observer.

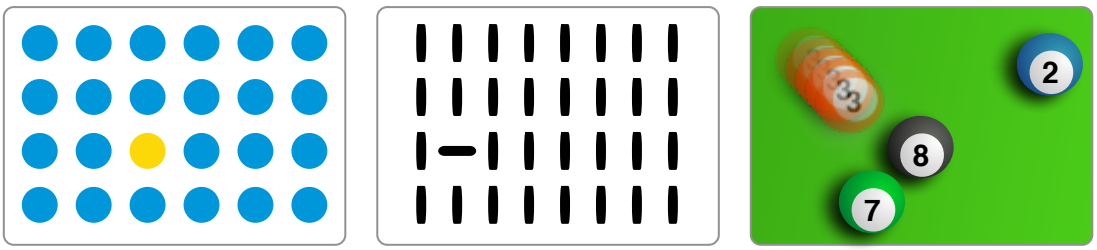


FIG. II.18 – L'extraction de la saillance doit être basée sur la notion d'information, plus une chose est rare, plus elle porte d'informations.

La figure II.17 illustre à travers un exemple comment l'agent peut choisir des régions de l'espace à visiter pour en extraire des représentations objectives sans en connaître au préalable

le contenu. Une carte de saillance met en avant des régions dont le contenu en terme de données brutes porte plus d'information au sens de Shannon[6]. Les régions d'intérêt peuvent être calculées sur la base de la forme, de la couleur, ou encore du mouvement (voir figure II.18) et de la stabilité de ces propriétés au cours du temps. D'une manière générale, l'agent doit être attiré par une propriété rare par rapport aux autres. La carte de saillance doit proposer une région de l'espace connexe et fournir au reste du système de représentation et au système d'action, l'information relative à cette région. La région proposée doit être unique à chaque instant.

L'agent ne doit pas observer la même région en permanence. Il doit explorer la scène visuelle de façon à en extraire le plus d'informations possible. De cette façon il aura une meilleure connaissance de son environnement et sera en mesure de savoir si un des objets qu'il observe possède une propriété capable de modifier son état subjectif. La carte de saillance doit donc être dynamique pour empêcher l'agent d'observer toujours la même propriété de l'espace. Elle constitue un moteur de focalisation de l'attention. Les maxima de saillance extraits doivent donc changer continuellement. Dans le cas de l'exemple présenté sur la figure II.17, les 3 maxima correspondent à une tasse (2) , une poire (1) et un réveil (3). La saccade suivante conduira l'agent à observer la tasse. Le moteur de saccade doit être tel que si tout le reste du système ne varie pas et que l'environnement reste identique, alors l'attention de l'agent se portera successivement sur ces 3 objets dans un ordre aléatoire défini par la fonction de saillance dynamique choisie.

Placer une région d'intérêt au centre de l'image peut se faire en utilisant une correspondance directe entre la région d'intérêt dans l'espace image et la variation d'angle nécessaire pour observer cette région. Nous ne formaliserons pas cette propriété, ce qui est important ici c'est de comprendre qu'il y a une correspondance directe entre la position des objets dans l'image et l'angle sous lequel la caméra observe ces régions.

II.7.4 Correspondance entre lieux et action : Représentation des lieux

Il existe une correspondance entre la position des objets dans le monde et les actions à produire pour interagir avec ceux-ci. L'agent peut donc représenter les lieux, c'est-à-dire la position des objets, sous la forme d'actions nécessaires à une interaction particulière. La nature des lieux que l'agent a besoin de manipuler est relative au comportement dans lequel ce lieu est engagé. Il y a autant de notions de lieux différentes que de façons d'interagir avec ces lieux. Dans le cas de la vision, la connaissance de la position d'un objet sera codée sous la forme d'un déplacement du capteur visuel pour centrer l'image de l'objet à observer. Dans le cas de la manipulation, la position terminale d'un bras par rapport à un objet peut constituer une autre forme de lieu, etc. D'une manière générale, cette correspondance entre position des objets et action peut être utilisée pour représenter les lieux. L'information relative à ces actions est perceptible à l'aide des capteurs proprioceptifs de l'agent. La correspondance entre ces différents lieux doit être apprise au cours de l'expérience et n'est pas donnée initialement.

[6]Les propriétés les plus informatives sont les plus rares

II.7.5 Propriétés nécessaires et suffisantes

Les Tableaux II.2 et II.3 situés en fin de chapitre dressent un bilan de l'ensemble des propriétés nécessaires et suffisantes que notre système de représentation objective doit posséder.

II.8 Apprentissage ouvert de fonctionnalités

L'agent est hédoniste par hypothèse. Ce que nous appelons fonctionnalité est la capacité de l'agent à créer des comportements qualitativement nouveaux et adaptés au fur et à mesure qu'il accroît sa capacité de se représenter l'environnement. Le moteur qui permet au système de toujours accroître le nombre de ces comportements et leur performance est le fait que chaque représentation objective possède une composante subjective (voir hypothèse II.3.2). L'agent cherche, en agissant, à accroître la perception de situations ayant une composante subjective positive. Afin de simplifier le raisonnement, nous allons introduire quelques notations sur les différents états dans lesquels le système peut potentiellement se trouver.

Par hypothèse, le sous-système de représentations subjectives est connecté à l'ensemble des représentations objectives. Le système de représentations subjectives est modélisé par un nombre réel[7] que nous l'appellerons $ES(t)$ pour état des représentations subjectives (Voir figure II.19, le fait que $ES(t)$ soit non nul est symbolisé par une paire de flèches sinusoïdales). Chaque situation rencontré va donc faire varier ce nombre. Nous avons vu que le système de représentation objectives était constitué d'unités de traitement organisées en graphe. Chaque unité de traitement possède une composante subjective telle que si l'unité s'active, elle contribuera à $ES(t)$. Initialement seules certaines représentations objectives relatives aux ressources internes de l'agent ont une influence positive ou négative sur ES. Les unités de traitement du système de représentation objective n'ont pas de composante subjective tant qu'elles ne sont pas différenciées, c'est-à-dire tant qu'elle n'ont pas appris à extraire une propriété particulière de l'environnement. Nous appellerons unités de récompense (UR) les unités de traitement dont l'influence sur l'état subjectif est positive et unités de punitions (UP) les unités de traitement dont l'influence sur ES est négative. Par extension nous appellerons état objectif récompense (EOR) les états objectifs dont l'état subjectif associé contribue à la croissance de ES et état objectif punition (EOP) les état objectifs dont l'état subjectif associé contribue à la décroissance de ES.

Initialement, l'agent atteindra son objectif, c'est-à-dire que $\frac{\partial ES(t)}{\partial t} > 0$, si il rencontre une situation qui influence positivement l'état de ses ressources internes (par exemple, si l'agent recouvre son niveau d'énergie maximum). De fait on peut considérer que l'objectif de l'agent est d'agir sur le monde pour activer ses unités de récompense et éviter d'activer les unités de punition.

Il existe plusieurs situations du monde qui lorsqu'elles sont associées à une action appropriée conduisent à activer ces unités de récompense (UR). Pour parvenir à atteindre son

[7] Les représentations subjectives pourraient avoir une représentation plus complexe, pour des raisons de simplicité nous considérons le cas d'un nombre réel.

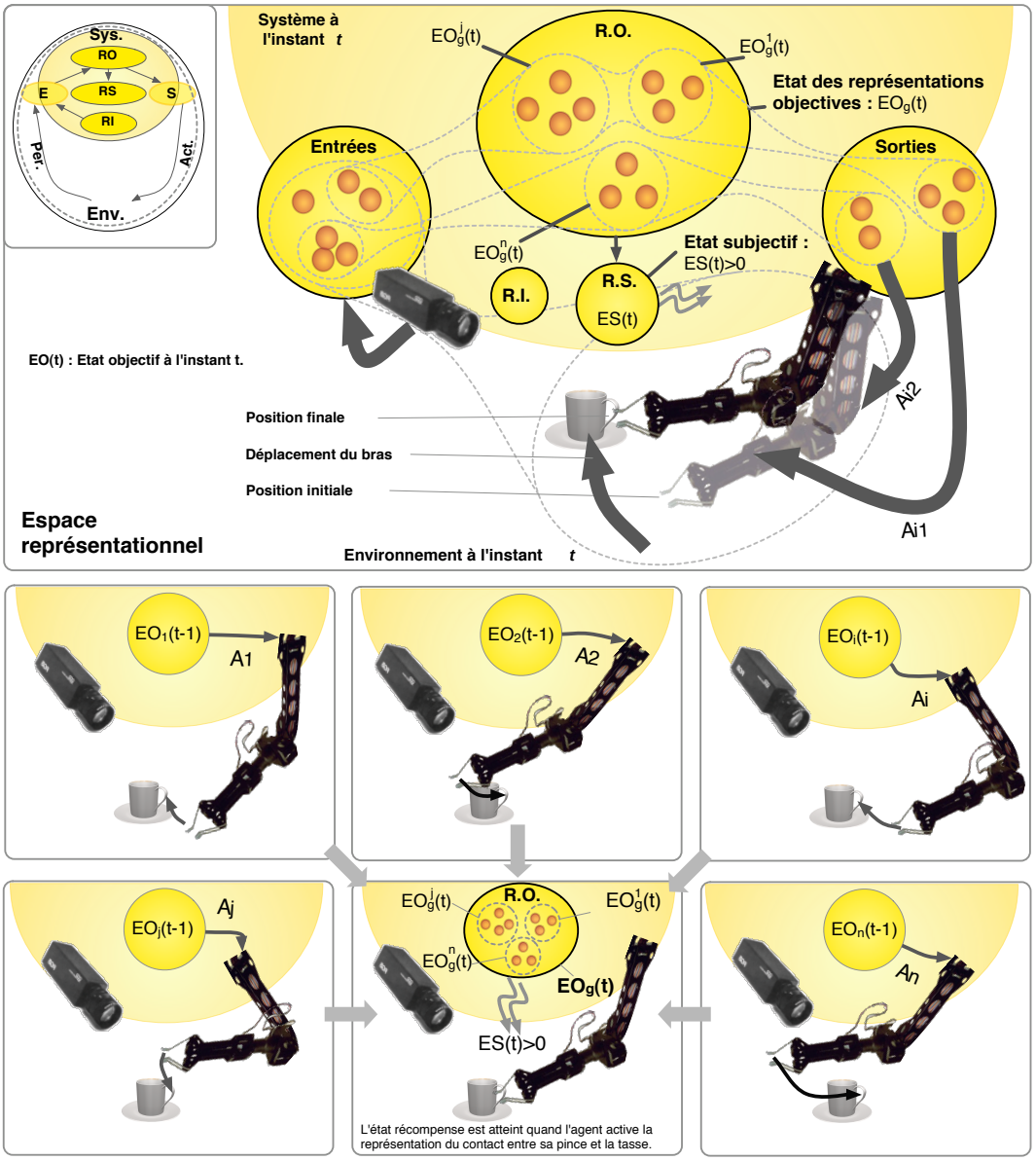


FIG. II.19 – Espace représentationnel.

objectif, l'agent doit donc apprendre à associer chaque états internes relatifs à ces situations avec une actions qui conduisent à une récompense. L'agent est autonome et n'a pas de connaissances *a priori* sur son environnement, il doit apprendre ces correspondances en expérimentant

son environnement. En faisant de nombreuses fois la même expérience, c'est-à-dire en faisant varier les conditions initiales et en atteignant le même EOR il sera en mesure de trouver les composantes objectives essentielles. Après quelques expériences, il devra avoir acquis suffisamment d'informations pour savoir quelles unités de traitement sont statistiquement causales avec son succès et quelles unités de traitement sont indépendantes de son succès. Pour mieux comprendre ce problème d'associations sensori-motrices nous allons introduire deux types d'espaces de représentations[8] : l'espace représentationnel et l'espace fonctionnel.

II.8.1 Espace représentationnel

L'espace représentationnel est celui que nous avons utilisé jusqu'à maintenant dans ce chapitre. Il décrit l'état du système à un instant donné. L'agent dispose d'un système de représentation ouvert. Chaque situation qu'il rencontre est décomposée en éléments indépendants relatifs aux différentes propriétés objectives. Nous appellerons état objectif à l'instant t, $EO(t)$, l'état d'activation de l'ensemble des unités de traitement. La figure II.19 détaille l'évolution de EO au voisinage d'un état récompense. Il est ici appelé $EO_g(t)$ parce qu'il constitue un but à très court terme pour l'agent. Cet exemple montre la variation de l'espace objectif et de l'environnement entre l'instant (t) ou l'agent atteint un état récompense et les instants ($t-1$) qui précèdent[9]. Ici l'état récompense est caractérisable dans l'environnement par le fait que la pince du bras de l'agent touche la tasse. L'agent modifie l'environnement et la configuration de son bras en produisant une action. Cette action est en fait la somme de plusieurs actions élémentaires. Nous discuterons de la représentation et de la synthèse de l'action plus loin. Nous supposons que l'agent est en mesure de se représenter cette situation de l'environnement et qu'elle conduit à un état objectif récompense. Naturellement, le fait qu'il s'agisse d'une expérience de manipulation guidée par la vision est un exemple qu'il est possible de généraliser. L'espace fonctionnel est pour cela plus adapté.

II.8.2 Espace fonctionnel

L'espace fonctionnel permet de représenter la dynamique de la coévolution des états représentations objectifs et subjectifs de l'agent. Nous dirons que deux états représentationnels sont voisins si l'agent dispose d'une action qui conduit d'une situation à l'autre. Cette notion de voisinage est orientée. Il n'y a pas nécessairement d'action inverse qui permette de retrouver l'état du monde précédent, par exemple si l'action consiste à briser un verre en le lâchant. La partie gauche de la figure II.20 montre l'ensemble des états voisins de l'état de récompense. On appellera orbite d'un état récompense $EOR(t_i)$ l'ensemble des états objectifs $EO(t_i-1)$ voisins de $EO(t_i)$. En toute rigueur chaque transition d'état objectif à lieu à des instants différents. Nous simplifions cette notation en remplaçant t_i par t. t indique également ici la notion de précédence relative, pas véritablement le temps.

[8] il s'agit ici de notre façon de représenter l'agent, pas des représentations de l'agent.

[9] t indique ici la notion de précédence relative, pas véritablement le temps.

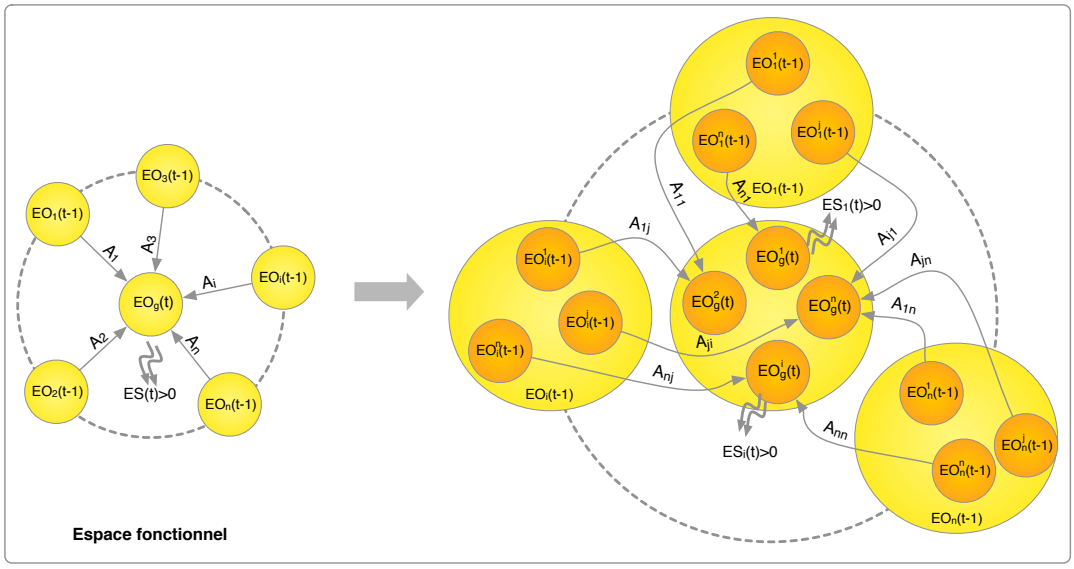


FIG. II.20 – Espace fonctionnel .

La partie droite de cette figure illustre le fait que chaque état de représentation objective est composé d'assemblées indépendantes chacune codant pour une partie de la situation et sa dynamique. Certaines assemblées d'unités de traitement doivent coder la tasse, l'anse, la taille de l'objet, la proximité de la pince, l'état proprioceptif du bras, etc. Cette indépendance est une condition nécessaire aux capacités d'apprentissage fonctionnel de l'agent. En multipliant les expériences de ce type et en modifiant l'objet à toucher, l'information relative à l'objet lui-même doit de moins en moins influencer le déplacement du bras. L'agent doit être en mesure de généraliser le fait que positionner sa pince en son centre de vision quel que soit l'objet qui s'y trouve permet d'atteindre un état récompense. Dans cet exemple, on voit que $EO_g^2(t)$ n'est pas impliqué dans l'obtention de la récompense, seul $EO_g^1(t)$ et $EO_g^i(t)$ le sont. $EO_g^2(t)$ peut, par exemple, être relatif à la couleur de la tasse. Lorsqu'il aura appris à ne retenir que les composantes des représentations causales avec son succès et qu'il aura associé tous les états de l'orbite de $EOR(t)$ avec les actions adéquates, il disposera d'une première fonctionnalité. Celle de placer sa pince au centre de l'image qu'il observe.

La partie inférieure de la figure II.19 illustre les différentes situations correspondant à l'orbite de l'état relatif au contact de la pince avec la tasse (EOR_g).

II.8.3 De l'action à la fonctionnalité

Nous nous plaçons dans le cas où il existe un état récompense et que le système évolue à son voisinage. Ce que nous appelons fonctionnalité de l'agent, c'est sa capacité à associer de

façon indépendante les composantes représentationnelles et les composantes d'action qui permettent d'atteindre un état récompense. Une fonctionnalité peut être vue comme une fonction qui va d'un sous-ensemble de composantes indépendantes dans un autre sous-ensemble par l'intermédiaire d'actions sur le monde.

L'agent doit apprendre ces associations sensori-motrices pour développer de nouvelles fonctionnalités. Initialement, il n'en dispose pas. Au cours de ses expériences, il doit créer des associations à partir des informations de causalité. Il sera dans un premier temps très maladroit, et peu à peu il va acquérir suffisamment d'informations pour affiner ses associations et détacher les sous-ensembles représentationnels et actifs indépendants. Une fois acquises, ces informations deviennent réutilisables dans d'autres schémas comportementaux. Pour revenir à notre exemple de la figure II.19 une fois que l'agent aura appris à positionner sa pince à une position donnée, il disposera de cette fonctionnalité pour en élaborer de plus complexes.

Dans l'exemple de la figure II.20, seuls $EO_i^n(t-1)$ et $EO_n^n(t-1)$ sont impliqués de façon causale dans l'activation de $EO_g^i(t)$, respectivement à l'aide de A_{nj} et A_{nn}. Cela signifie que les autres sous-états des différents états de l'orbite ont totalement perdu leur influence dans cette activation et le fait que $ES_i(t) > 0$. Ce type de cas correspond à un système ayant déjà appris une fonctionnalité.

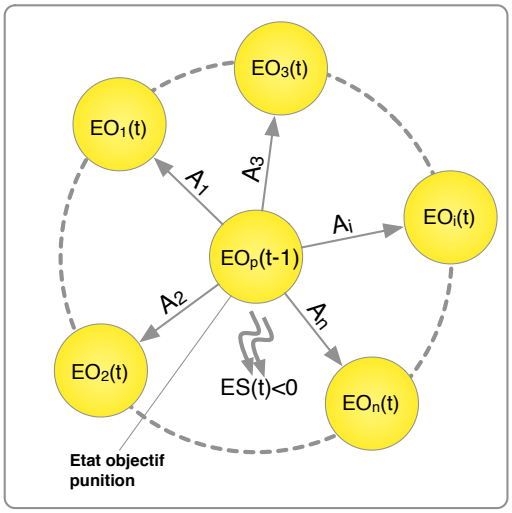


FIG. II.21 – Espace fonctionnel, cas d'un état punition.

L'évitement d'états punitions (voir figure II.21) permet à l'agent de rendre inactives des composantes représentationnelles de punition. Cet évitement ne permet pas de construire une fonctionnalité dans la mesure où il ne constitue pas un point de convergence de l'orbite vers le centre mais un point de divergence du centre vers l'orbite. L'agent peut par contre réutiliser ses fonctionnalités pour éviter un sous-état de punition.

La complexité de l'environnement permet d'avoir dans une même situation à la fois des sous composantes représentationnelles de punition et de récompense. Le fait que les fonctionnalités soient uniquement dépendantes d'une sous-partie de l'état représentation objectif permet, s'il n'y a pas de conflit matériel, de combiner plusieurs fonctionnalités simultanément. Par exemple le fait d'approcher une pince de l'objet et en retirer une seconde simultanément.

Un état récompense peut être considéré par le système comme un but. C'est en expérimentant son environnement au voisinage de l'état récompense que l'agent peut développer une fonctionnalité. Nous savons aussi par hypothèse qu'initialement, seul certains états relatifs aux ressources internes ont une composante subjective. Comment faire en sorte que l'agent développe de nouvelles fonctionnalités qui ne soient pas relatives aux représentations objectives des ressources internes ?

II.8.4 Apprentissage de la composante subjective

Pour que l'agent puisse apprendre de nouvelles fonctionnalités, il faut qu'il dispose de nouveaux états objectifs de récompense. Quelque soit la situation rencontrée par l'agent, il existe un état objectif et subjectif. L'état subjectif global est relatif aux composantes subjectives des différentes unités de traitement objectives actives à cet instant. Si l'agent vient d'agir, alors l'état subjectif global est causal, au moins en partie de l'état subjectif dans lequel il se trouve.

Intuitivement on comprend que les états voisins de l'état de récompense doivent être eux aussi des états de récompense. Si l'agent a appris toutes les correspondances entre états d'une orbite et actions pour atteindre l'état récompense, alors agir pour atteindre cette orbite constitue un premier pas vers la récompense. Ceci n'est possible que si les états de l'orbite sont récompensants. Pour qu'ils le soient, il suffit d'attribuer à ces états une part de l'état récompense central. Pour cela il suffit de propager cette valeur aux différents états de l'orbite. Au cours des différentes expériences de l'agent, tout état qui précédera un état récompense se verra attribuer une part de cette récompense. Rappelons qu'un état objectif est constitué par l'ensemble des unités de traitement actives et inactives. Par hypothèse, seules les unités actives contribuent à l'état subjectif global. Ce sont donc les unités qui composent l'état précédent qui vont hériter de l'état subjectif global central. L'orbite doit devenir un attracteur. Il doit exister une voie qui va du système de représentations subjectives vers le système de représentations objectives. La figure II.22 illustre le développement de ce réseau d'orbites. Il faut garder à l'esprit qu'il existe en réalité plusieurs sous réseaux d'orbites correspondants chacun aux différentes composantes indépendantes des états objectifs de l'agent.

Nous avons vu que les représentations devaient être organisées en graphe et de façon hiérarchique. On comprend aisément que les unités de traitement trop proches des entrées ne peuvent avoir une composante subjective très élevée. En effet ces représentations élémentaires sont utilisées dans de nombreuses représentations. Plus une unité de traitement est éloignée des données brutes au sens du graphe, plus elle représente une propriété découplée d'une configuration particulière de ces données brutes. Lorsqu'une unité est loin des entrées, son activité traduit la présence d'une catégorie de propriété. La figure II.23 illustre cette propriété. Supposons que l'agent rencontre un état récompense et qu'il observe simultanément une pomme. Supposons

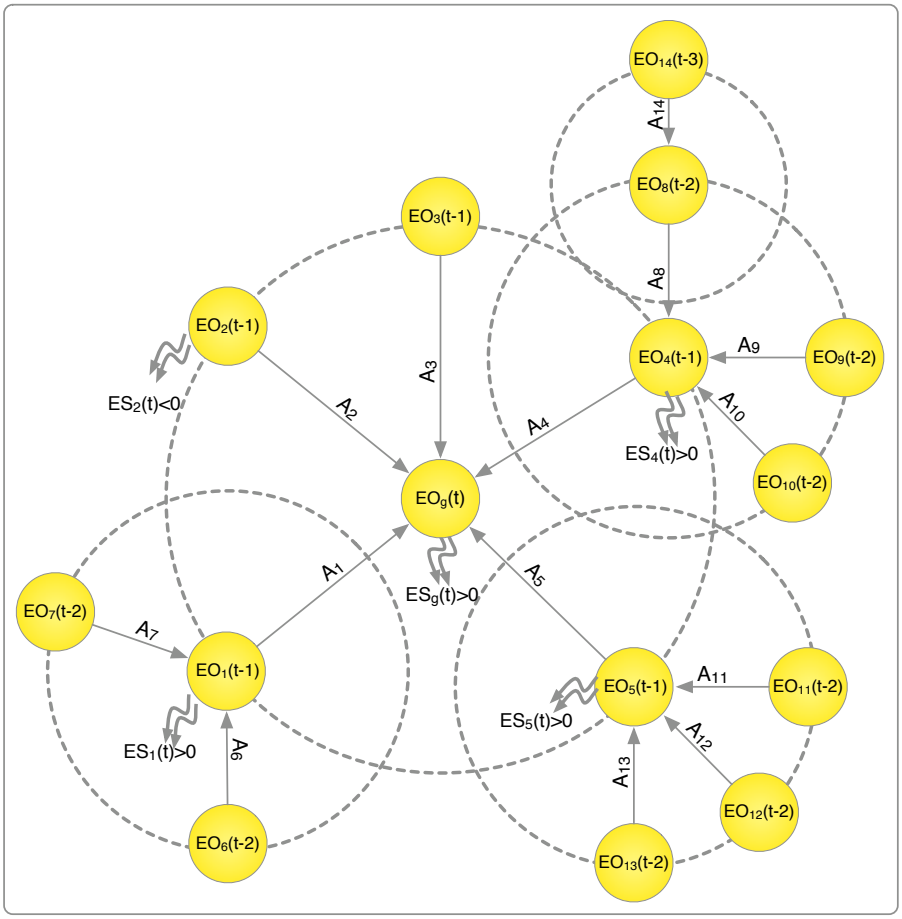


FIG. II.22 – Exemple de fonctionnalité à plusieurs orbites. Il faut garder à l'esprit que chaque état objectif $EO(t)$ est la superposition d'états indépendants se trouvant chacun sur des orbites indépendantes.

qu'il ait une assemblée d'unités de traitement codant pour la couleur rouge, une autre pour une forme arrondi et enfin une dernière pour le fait qu'il y ait une pomme et qui dépende des deux premières assemblées. Si l'agent atteint systématiquement un état récompense quand il observe une pomme, mais que par contre il n'en obtient pas lorsqu'il observe simplement un objet rouge seul (ici une fleur, ou un objet arrondie seul, ici un ballon) alors l'assemblée codant pour la pomme héritera de la propriété de récompense alors que l'assemblée codant pour le rouge et celle codant pour la forme arrondie n'apporteront finalement pas de récompense. A la suite de cette expérience, l'agent dans ce cas cherchera des objets rouges ou arrondis ou des pommes, et peu à peu, il ne cherchera plus que des pommes.

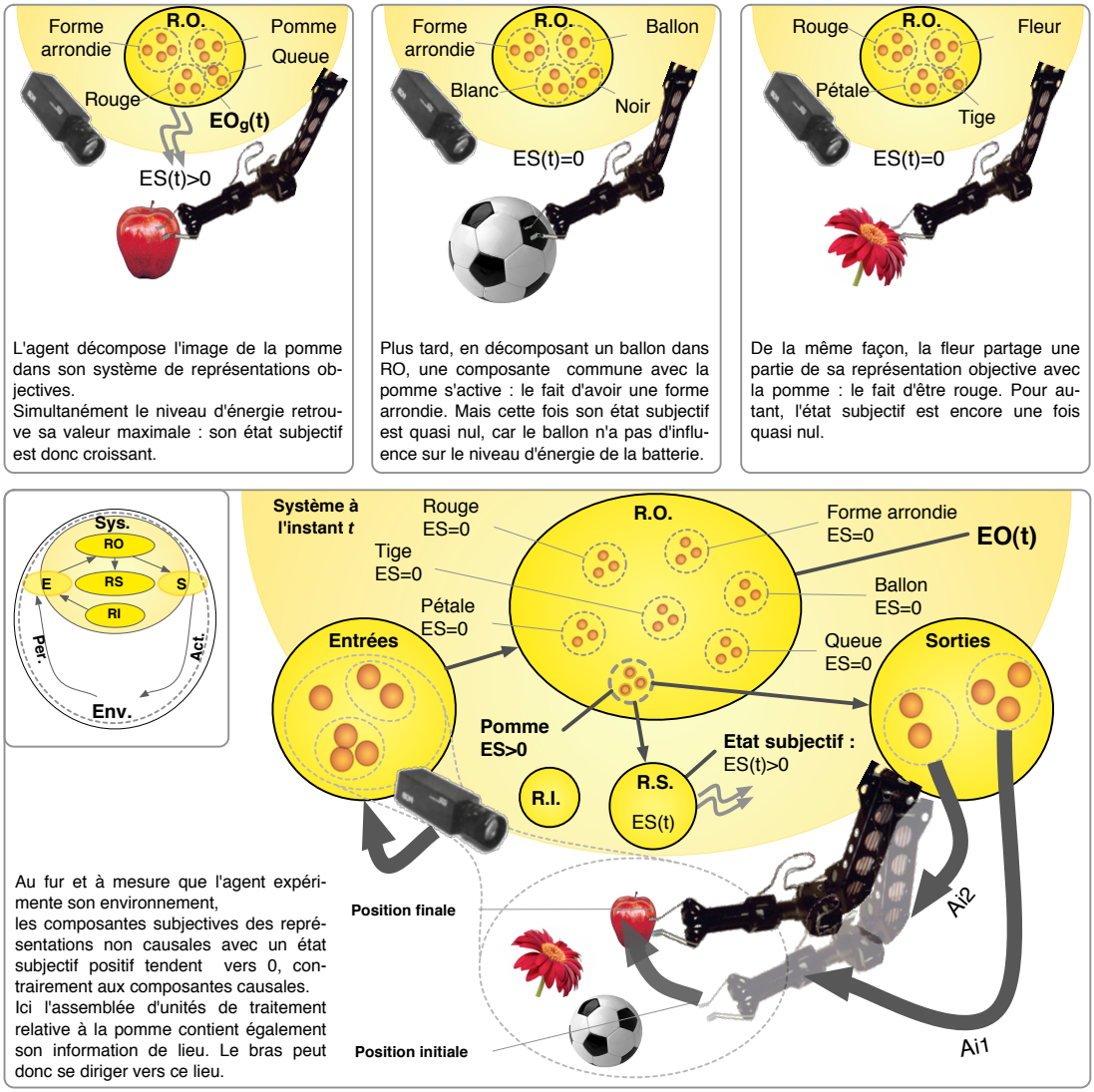


FIG. II.23 – Exemple de propagation de la composante subjective.

II.8.5 Résolution de la perception et de l'action

Les variations de l'environnement susceptibles d'être la cause des variations de son état subjectif peuvent prendre n'importe quelle forme, n'importe quelle taille, avoir n'importe quelle durée ou être situées n'importe où. Pour maximiser ses chances de détecter ces causes, il est nécessaire que l'agent observe le plus d'informations simultanément. Il semble moins utile de justifier une perception multiéchelle qu'une perception à une échelle unique.

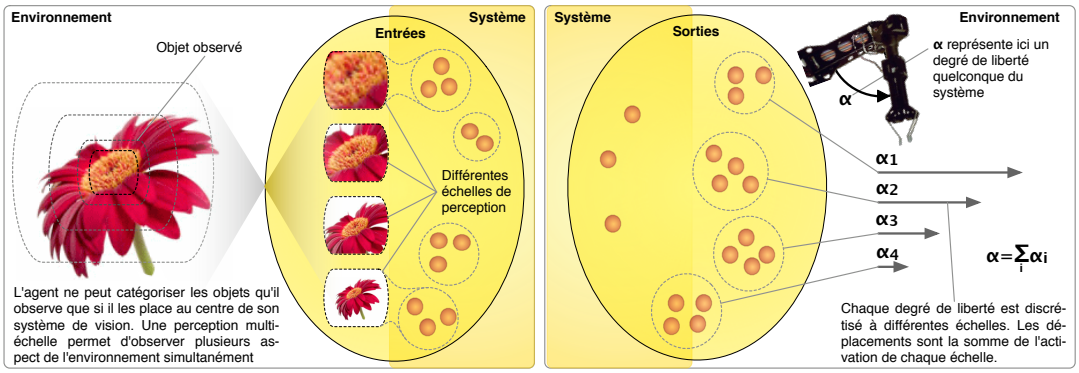


FIG. II.24 – Perception et action multi-échelle.

L'agent peut observer le monde à l'aide d'un grand nombre de capteurs. Un capteur visuel lui donnera des images des différents objets qui l'entourent. Dans une image, les objets ont une extension spatiale et temporelle, ils ont une taille et une position. Un capteur auditif lui donnera un signal n'ayant pas de dimension spatiale, mais une extension spectrale et une durée. Deux capteurs auditifs placés de façon adéquate permettront d'obtenir de l'information sur la position de la source sonore. Deux capteurs visuels placés côte à côte, permettront d'extraire de l'information de profondeur et de connaître la distance relative à laquelle se trouvent les objets observés. Observer le monde à toutes les échelles spatiales ne suffit pas. L'agent est situé, il est quelque part dans le monde, en cela, il a un point de vue particulier sur son environnement. Il ne peut percevoir qu'une partie de ce dernier, son point de vue est partiel. Pour pallier ce problème, il doit pouvoir changer de point de vue sur son environnement. Il doit pouvoir se déplacer, ou déplacer ses capteurs pour augmenter ses chances de trouver les causes des variations de son état subjectif. Cette propriété de mobilité est nécessaire pour observer les propriétés de l'environnement qui varient en fonction de l'endroit où l'agent se trouve pour les observer.

L'agent doit également agir pour retrouver des situations qui vont accroître son état subjectif. Les actions qu'il peut avoir sur le monde dépendent de ses effecteurs. Quels que soient ces effecteurs, elles peuvent se résumer d'un point de vue représentationnel à un ensemble de degrés de liberté. Dans leur expression la plus simple, il peut s'agir d'un simple interrupteur binaire. Les actions les plus complexes peuvent être considérées comme la somme de variations de degrés de liberté simples. Nous avons vu que l'agent représente son environnement de façon totalement distribuée à travers l'activité d'unités de traitement dont le sens est donné par la position dans le graphe. Il n'y a aucun moyen de connecter des unités de traitement dont la sémantique est initialement inconnue à des lois de commande particulières. Par contre, il est possible de connecter des unités particulières du réseau à des lois de commandes préfabriquées et de laisser l'agent connecter les bonnes représentations aux bonnes actions au cours de son expérience. Chaque degré de liberté est susceptible d'être commandé en vitesse,

en amplitude, ou pourquoi pas en accélération. Une seule unité de traitement ne suffirait pas à manipuler l'ensemble des paramètres de commande d'un degré de liberté. L'agent ne connaît pas initialement les actions qu'il devra accomplir pour causer une variation positive de son état subjectif. Il peut s'agir dans les cas les plus simples d'amener une pince à une position donnée. L'agent doit trouver la bonne commande à appliquer à chacun de ses degrés de liberté pour maximiser ses chances de trouver l'action la plus appropriée à l'accroissement de son état subjectif. De la même façon que la représentation de la perception, la représentation de la dynamique des degrés de liberté est choisie comme étant multiéchelle. La dynamique de chaque degré de liberté est prise en charge par plusieurs unités de traitement. Certaines unités codent pour des variations d'amplitude, d'autres pour des variations de vitesses. Chacun de ces paramètres est codé de façon multi-échelle par un ensemble d'unités, chaque unité pilotant une échelle particulière d'un paramètre donné. Cette méthode permet de représenter efficacement et indépendamment la vitesse et l'amplitude de chaque degré de liberté.

Conjuguée à une perception multiéchelle, cette propriété prend tout son intérêt. L'agent peut expérimenter ses effecteurs en activant aléatoirement les unités de traitement relatives à chaque degré de liberté. Très vite, il peut trouver qu'il existe une échelle spatiale privilégiée pour commander un degré de liberté donné et que les autres ne sont qu'une suite de facteurs correctifs. Si on considère l'exemple d'un agent muni d'un bras et d'une caméra. En supposant également qu'il dispose de deux degrés de liberté pour déplacer sa pince, un pour les déplacements de droite à gauche et un autre pour les déplacements de haut en bas. Le fait de disposer une perception multiéchelle permet à l'agent de percevoir grossièrement dans quel quart de l'espace se trouve sa pince. Le fait de pouvoir agir à différentes échelles sur ces degrés de liberté lui assure qu'avec un minimum d'itération il pourra apprendre à associer la perception de son bras dans le bon quart d'espace avec le déplacement de grande amplitude approprié. Une fois le problème grossièrement résolu, l'indépendance des échelles spatiale et la propriété de composition des actions permettent d'utiliser cette solution partielle comme base d'action et d'ajouter progressivement des composantes plus fines pour augmenter la précision de l'action. Toutes les actions que l'agent est susceptible d'apprendre à accomplir peuvent se ramener à une composition d'activation d'unité codant pour la dynamique des degrés de liberté. Il s'agit du même processus de recherche d'association quelque soit les échelles d'actions et de perceptions considérées. En composant l'action résultante comme la somme des toutes les actions à toutes les échelles, le système dispose d'un système efficace d'apprentissage action.

La figure II.24 illustre les propriétés multiéchelle à la fois perceptive et active de l'agent. La qualité et le nombre des capteurs et d'effecteurs que l'agent va utiliser est une donnée initiale. Les capteurs et les effecteurs doivent seulement être choisis de telle sorte que l'agent puisse percevoir les variations de l'environnement qu'il a occasionnées en agissant.

Revenons sur la première expérience (II.4.2), celle de l'agent muni d'une caméra, d'un haut-parleur et de deux boutons de récompense. L'action de l'agent est dans ce cas précis la capacité d'émettre un "bip". Son retour est la perception directe d'une récompense ou d'une punition. Son action est binaire, sa perception conjuguée l'est aussi. Il est parfaitement adapté parce que la complexité de son action et la complexité de sa perception sont en accord. La complexité du traitement est transférée à l'expérimentateur qui doit choisir les images cibles,

les reconnaître et appuyer sur le bouton de récompense ou de punition.

II.8.6 Optimisation des comportements

L'agent va expérimenter son environnement en explorant les possibilités que lui offrent ses effecteurs. Cette façon d'apprendre à réaliser des actions appropriées pose un problème. Une solution trouvée en utilisant une méthode d'expiration peut conduire à beaucoup de solutions. Certaines étant plus coûteuses que d'autres en énergie.

Supposons par exemple que l'agent possède deux bras et qu'il obtienne un accroissement de son état subjectif quand la terminaison de son bras droit se trouve à la proximité d'un objet particulier. S'il apprend à agir en activant aléatoirement les différentes composantes de ces degrés de liberté, il existe des actions solution où le bras gauche se déplace aussi pendant l'action. Or dans ce cas précis le déplacement du bras gauche est inutile à l'obtention de l'accroissement d'état subjectif. Ces solutions particulières ne sont pas optimales au moins en terme d'énergie dépensée.

Une condition suffisante permet au système de supprimer les composantes d'actions inutiles. Elle consiste à associer l'usage des effecteurs avec la perception de la perte d'une ressource interne, c'est-à-dire d'associer chaque modification des effecteurs avec une décroissance de l'état subjectif. Souvenons-nous que l'objectif de l'agent n'est pas d'avoir un fort état subjectif, mais que cet état subjectif soit croissant. S'il trouve une solution à son problème, il va continuer à chercher à accroître son état subjectif. Souvenons-nous également que ses actions sont multiéchelle (II.8.5). Chaque action est la somme de composantes apprises indépendamment elle-mêmes connectées à des représentations indépendantes. La seule façon pour l'agent de continuer à accroître son état subjectif est de retirer les composantes inutiles progressivement et au cours de son exploration, autrement dit les composantes qui ne participent pas au succès de l'action. Il faut faire attention à ce que l'action optimale soit moins coûteuse que la récompense.

II.8.7 Sémantique et contexte

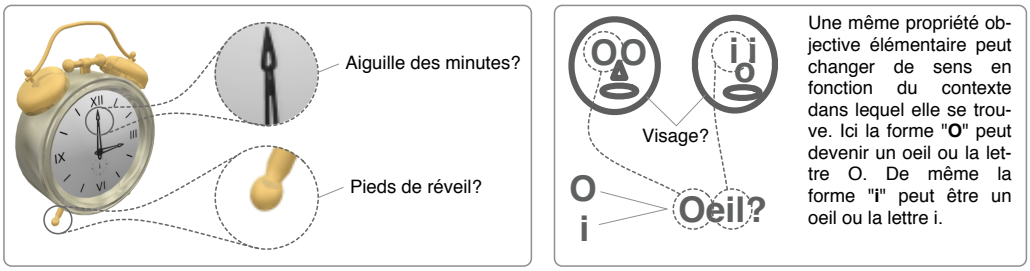


FIG. II.25 – Importance du contexte.

Le contexte dans lequel est observé une propriété de l'environnement peut parfois jouer une rôle très important. Il peut arriver qu'une même propriété objective prenne un sens totalement différent en fonction des propriétés objectives qui l'entourent. La figure II.25 met en évidence ce type de situation. Si on s'intéresse à la partie gauche de la figure on peut se demander si l'aiguille des minutes du réveil n'est pas un pinceau. La partie droite montre qu'une forme géométrique simple peut être interprétée comme un oeil ou comme une lettre selon son contexte. Il n'y a pourtant aucune différence objective locale. La propriété de hiérarchie (II.4.7) permet d'envisager qu'une assemblée d'unités de traitement de plus haut niveau utilise des propriétés géométriques de plus bas niveau pour s'activer. Le sens associé à une forme simple n'est donc pas attribué dès que cette forme est détectée par l'activation d'une unité de traitement. Pour que cette propriété objective prenne un sens, il est nécessaire que ce traitement soit fait plus haut dans le réseau. Le sens d'une propriété objective ne peut se trouver dans cette propriété elle-même. En revanche, cette propriété est nécessaire pour que d'autres unités puissent lui donner un sens. La propriété permet d'avoir un sens codé ailleurs dans le réseau et si les unités qui codent pour ce sens sont en phase avec l'unité codant pour une propriété objective simple, alors elle héritera de ce sens là. De la même façon qu'elle est associée à un lieu, une taille, une couleur, etc... Il est probable que les échelles spatiales basses permettent de dégrossir le problème, puisqu'elles permettent d'observer les scènes de façon globale et contiennent l'intégralité du contexte. Le contexte permet de mettre en évidence que certaines perceptions sont émergentes, dans le sens où elles n'existent que grâce à la contingence de plusieurs propriétés ensemble et n'existent pas si ces unités son considérées séparément, c'est un argument supplémentaire en faveur de la propriété de hiérarchie (II.4.7).

La nécessité d'apprendre la composante subjective des unités de traitement (II.8.4) met également en évidence que les unités de bas niveau ne peuvent hériter d'une composante subjective très élevée car elles peuvent prendre part à plusieurs schémas perceptifs. Il est donc nécessaire qu'une assemblée de traitement de plus haut niveau puisse hériter de cette composante subjective. Ceci constitue un nouvel argument en faveur d'un système de représentation hiérarchique.

II.9 Discussion

Nous avons jusqu'à présent donné une liste de conditions parfois nécessaires, parfois suffisantes, pour qu'un agent hédoniste tel qu'il est défini dans nos hypothèses accroisse ses connaissances sur le mode et le nombre de ses fonctionnalités. Nous allons maintenant discuter ces conditions et essayer de les analyser sous un angle différent.

II.9.1 Stockage et traitement de l'information : Mémoire et calcul

Comment un tel système traite et stocke l'information ? Les propriétés de fidélité, continuité et hiérarchie assurent que lorsque l'agent a appris à décomposer un objet dans son système perceptif, la chaîne causale relative à la propagation d'une information perceptive suivra tou-

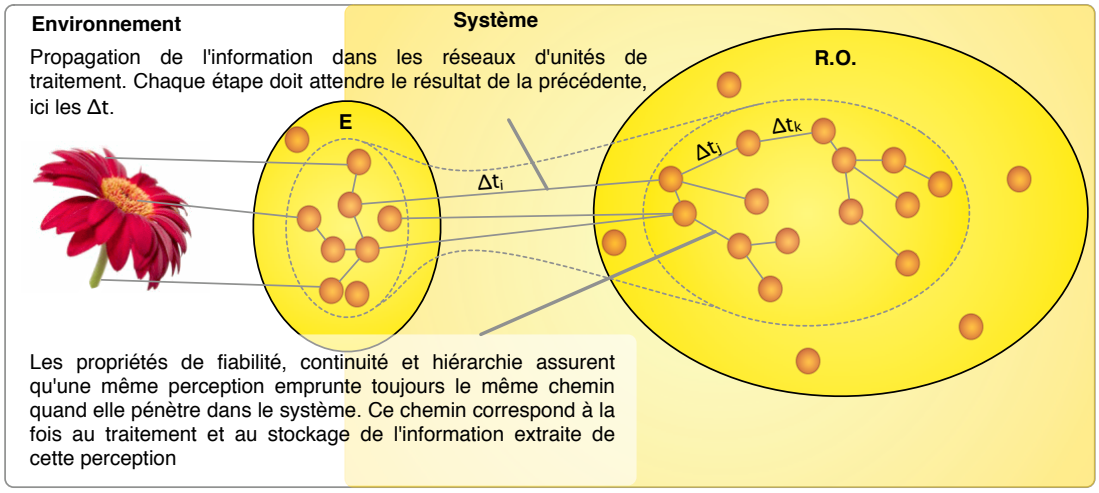


FIG. II.26 – Stockage et traitement de l'information.

jours le même chemin. Ce chemin est précisément le lieu où va être traité et stocké l'information relative à cette perception. C'est ce qu'illustre précisément la figure II.26.

Chaque perception va suivre son propre chemin. Ce chemin prend la forme d'un arbre dont les branches correspondent à l'extraction de plusieurs propriétés indépendantes dans une même assemblée de traitement. Le fait de coder les propriétés de l'environnement de façon distribuée consiste à extraire simultanément plusieurs propriétés relativement aux propriétés physiques d'un même espace conjugué. Si deux perceptions ont un point commun, alors elles auront une partie de l'arbre de perception en commun. La propriété de superposabilité des états permet ce cas de figure. Le sens de l'activation d'une unité de traitement est donné par sa position dans le réseau. Chaque unité de traitement en observe d'autres. Le sens qu'elle véhicule dépend du sens de l'ensemble des unités qu'elle observe.

Deux perceptions différentes conduiront à des arbres différents. Si ces deux perceptions ont une propriété commune, alors une partie de leur arbre sera commune.

C'est en décomposant les informations brutes que l'arbre va progressivement s'activer. En se propageant, l'information active des unités de plus en plus élevées dans le système. Plus ces propriétés sont élevées, plus elles doivent rendre compte de propriété stables (II.5.1, II.5.2). Le rôle de la perception pour ce type d'agent consiste à caractériser l'environnement en un ensemble de propriétés simultanément présentes, et qui soient les plus stables et les plus indépendantes possibles. Plus une propriété est stable, plus elle va persister quand les données brutes issues des capteurs vont varier. L'agent doit seulement catégoriser son environnement et apprendre à agir en fonction de ces catégories. L'information est traitée tout au long de la propagation, et elle est stockée sur toute la longueur de l'arbre. Chaque unité activée informe potentiellement le reste du système qu'une catégorie a été détectée.

II.9.2 Accès aux représentations pour un observateur

Une représentation dans un système d'IA symbolique est directement accessible à un observateur. Elle est créée par un concepteur et il est possible à chaque instant de suivre explicitement chaque représentation du monde, c'est-à-dire les valeurs relatives à chaque symbole. Le concepteur sait où trouver une variable relative à un paramètre de son modèle dans un programme. Le codage par position et le fait que l'agent puisse créer de nouvelles représentations de façon autonome posent la question de l'accès aux représentations par un observateur. Si l'agent crée ses représentations, il choisit seul l'endroit où il va les stocker. Les informations que l'agent manipule lui sont propres. Si une unité de traitement se spécialise et apprend à coder une catégorie particulière, alors l'information que véhicule son activation devient disponible pour les unités qui l'observent. L'agent est seul à savoir trouver les unités qui codent pour des propriétés. Ces informations ne sont pas directement accessibles pour un observateur qui étudierait le fonctionnement du système (voir figure II.26). Il est le plus souvent impossible d'identifier de façon formelle le type de propriété qu'une unité de traitement est capable d'extraire. Le fait que cette information ne soit pas accessible à un observateur extérieur au système n'est pas incompatible avec notre objectif. Le fait qu'il ne soit pas facile d'accéder à cette information n'est pas incompatible non plus avec le fait que l'agent puisse appendre de nouvelles représentations et de nouvelles fonctionnalités.

II.9.3 Ouverture et homogénéité du traitement

Ce type d'approche est ouvert. Le fait de représenter le monde en utilisant un codage par position permet de disposer d'un système capable de s'adapter à des problèmes très différents. L'ouverture de la fonctionnalité est héritée de l'ouverture du système de représentation. Le sens relatif à l'activation des unités de traitement est donné par la position de cette unité dans le réseau. Si un système dispose de ce type de représentation, il peut *a priori* représenter n'importe quelle cause de n'importe quel accroissement de son état subjectif pour peu qu'il dispose de suffisamment d'unités de traitement.

Il est intéressant de faire un parallèle avec un autre système ouvert bien connu : le système de numération par position. Le statut de l'unité de traitement peut ici être comparé au statut du chiffre. Le statut des règles de calcul élémentaire (addition, soustraction, multiplication division) correspond ici aux règles locales élémentaires mettant en jeu des unités de traitement. Il est intéressant de noter que ces règles sont totalement indépendantes du type de données qu'elles sont supposées traiter et ne porte que sur la position relative et l'état des unités de traitement dans le réseau.

Contrairement aux approches symboliques, le sens ne se met pas en travers de l'accroissement des connaissances du système. Il n'est pas nécessaire de construire des modèles particuliers pour chaque problème posé. Plutôt que de développer une théorie pour chaque problème posé, ce système ouvert de représentations et de fonctionnalités permet d'exprimer n'importe quelle classe de situations pour peu qu'il y ait suffisamment d'unités de traitement.

Le système de numération par position permet d'exprimer tous les nombres et de s'appro-

cher infiniment près de certains, car il fait abstraction de la notion de sens. Le lieu devient le sens et un raisonnement sur ce lieu permet de réaliser des calculs. De la même façon ce système de représentation permet d'exprimer n'importe quelle classe de tâche sensori-motrice, même celles que nous ignorons en tant que développeur d'un tel système. Chaque système pris individuellement aura une limite dans son développement, en particulier sa capacité à catégoriser des propriétés de plus en plus générales. En poursuivant le parallèle avec le système de numération par position, on peut considérer que pour un niveau de catégorisation nécessaire, il existera un système de représentation objective distribué satisfaisant, de la même façon que le système de numération par position permet de coder des quantités arbitrairement grandes et d'effectuer avec n'importe quel nombre n'importe quelle opération en ne se souciant que des valeurs des chiffres localement et jamais du nombre lui-même.

D'un point de vue modélisation, ce type de système doit mettre en oeuvre un ensemble fini d'algorithmes simples et mettre à jour en permanence les données relatives à l'état des unités de traitement. Ce type de traitement de l'information est homogène, il est le même partout. Quelle que soit la partie du système qu'on observe, elle sera composée des mêmes unités de traitement. Nous devrons donc définir un ensemble réduit d'algorithmes et les combiner spatialement pour coder les représentations, et temporellement pour coder les traitements. Ces algorithmes élémentaires sont l'équivalent des chiffres. Les règles locales qui régissent l'évolution de l'état ont le statut des opérations arithmétiques.

Il est intéressant de noter que les systèmes ouverts connus respectent cette propriété de codage par position. Le système de numération indien est sans doute le système ouvert artificiel le plus ancien. Il existe aussi des systèmes ouverts naturels. Le cerveau des primates est constitué de neurones. Il en existe une vingtaine de classes anatomiquement différentes. L'activation d'un neurone est fonction de sa position : une cellule ganglionnaire code un contraste local positif ou négatif ; une cellule du cortex visuel primaire code des orientations locales, des disparités oculaires, des mouvements ou des couleurs ; une cellule du cortex inféro-temporal peut coder des visages, etc. Les règles de fonctionnement portent sur l'état des neurones et pas sur le sens relatif à leurs décharges. En cela les neurones possèdent les mêmes propriétés d'ouverture que le système de numération indien. D'une manière générale, les cellules vivantes partagent aussi cette propriété. Initialement semblables, leur différenciation en cellules d'os, de peau, en neurones, etc. est fonction de leur position relative. Enfin, la matière est composée d'un ensemble fini de particules élémentaires et les lois de la physique portent sur la nature et la position des particules.

Les catégories objectives d'un objet physique résident dans sa forme dans un sens large. Cette forme est due à la position des particules qui le composent dans l'espace. S'il code lui-même le monde avec un tel code, alors sa perception est une fonction qui va d'un système de codage par position vers un autre.

II.9.4 Parallèle avec les neurosciences

Ce chapitre est une analyse des conditions nécessaires et parfois suffisantes à un système d'apprentissage ouvert de représentations et de fonctionnalités. Notre hypothèse de départ

considère ce problème indépendamment des travaux menés en neurosciences ces dernières années. Nous pensons que si une machine peut posséder des capacités de généralisation de ses actions en fonction d'un contexte perceptif équivalentes à celles de l'homme, elle n'en est pas pour autant une copie. Nous pensons qu'il existe une classe de système définit par l'ensemble des propriétés que nous avons exhibées, et que ces propriétés peuvent être naturellement mises en oeuvre dans le cerveau ou dans un système robotique.

On retrouve beaucoup de travaux en neurosciences et en psychologie cognitive allant dans le sens des propriétés nécessaires exhibées dans ce chapitre. Les propriétés suffisantes quant à elles ont été inspirées de théories issues des neurosciences et de psychophysique.

Représentations distribuées et codage par position

La manière dont l'information est représentée par l'activité des neurones est une question essentielle en neuroscience. Il existe plusieurs théories à ce sujet, dont la théorie dite de la cellule grand-mère qui considère qu'un neurone seul permet de coder objet [Barlow 72] ou celle qui considère que les représentations sont distribuées dans l'activation d'assemblées de neurones. La théorie de la cellule grand-mère pose un problème : elle ne permet pas de généraliser la production des actions (voir section II.4.2). Nous considérons que les représentations sont distribuées.

La première des propriétés est peut-être la notion de codage par position. Il n'existe que quelques types de neurones chez le primate (cellule multipolaire, pyramidale, cellule de Purkinje, cellule bipolaire, cellule monopolaire, etc.) [Kandel 91]. Les cellules ganglionnaires dans la rétine extraient différents types de contrastes, les neurones de V1 extraient des orientations locales, des disparités, des directions de mouvement, ou encore des couleurs. Certains neurones de IT extraient des visages vus de face, etc. Les neurones communiquent des informations en s'échangeant des potentiels d'action. Il semble que la sémantique véhiculée par l'activation des neurones soit une fonction de leur position dans le cerveau. Ces données vont dans le sens d'un codage par position tel que nous l'avons décrit à la section II.4.8. Ce que nous appelons unité de traitement se rapproche plus de la notion de colonne corticale que de la notion de neurone (voir chaitre III).

La voie du « où » et la voie du « quoi » : vers des représentations totalement distribuées"

Schneider [Schneider 69] a montré que des hamsters dont une partie du cortex visuel avait été lésée étaient incapables d'apprendre de nouveaux motifs visuels, mais étaient toujours en mesure d'apprendre de nouvelles propriétés spatiales. En 1982, Mishkin et Ungerleider [Mishkin 82], [Ungerleider 82] ont obtenu des résultats similaires chez le singe, en particulier le fait que le système visuel des primates se divise en deux voies principales : la voie dorsale et la voie ventrale. Chacune de ces voies est spécialisée dans un aspect particulier de la vision. Chacune de ces voies part du cortex occipital, en particulier l'aire V1. La voie ventrale se prolonge jusqu'au cortex inféro-temporal et traite la question du « quoi », la voie dorsale se termine dans le cortex supérieur pariétal et traite de la question du « où ».

À la suite d'une lésion de la voie ventrale, les singes étaient incapables de reconnaître les objets qui leur étaient présentés, mais étaient capables de les localiser. Ceux ayant la voie dorsale lésée étaient capables de reconnaître les objets, mais pas de les localiser. De tels résultats ont également été montrés chez l'homme [Levine 85],[Ungerleider 94].

La condition portant sur la distribution des représentations section II.4.3 est bien compatible avec les résultats de ces travaux. Un système de représentation totalement distribué permet de trouver de l'information commune entre deux contextes perceptifs différents. Représenter l'environnement de cette façon confère au système des propriétés de généralisation des actions. La section II.7.2 est une condition qui met en évidence qu'il n'est pas judicieux que le système possède des unités codant à la fois la forme et le lieu. Si tel était le cas, chaque nouvelle forme serait associée à une notion de lieu particulière, et chaque forme inconnue nécessiterait un nouvel apprentissage de la notion de lieu associée. Un tel système ne serait pas en mesure de généraliser la notion de lieu une fois pour toutes indépendamment de la forme. Notre analyse conduit donc à la même conclusion que celle proposée par Mishkin et Ungerleider chez le singe : L'information de forme et l'information de lieu doivent bien être séparées.

Oscillations et synchronie des assemblées de neurones

La section II.6.1 met en évidence qu'un système de représentation totalement distribué est ambigüe. Si le système se trouve confronté à 2 objets spatialement séparés, l'information liant chaque représentation de chaque forme à une représentation de position est perdue. Ce problème est également valable pour tous les attributs qui caractérisent un objet ou une partie de l'environnement (couleur, taille, orientation, etc.).

L'ensemble des aires visuelles du primate traite l'information de façon distribuée et se trouve également confronté à ce problème. En 1989, Gray a montré que chez le chat, des neurones spatialement séparés peuvent avoir une activité synchrone [Gray 89]. Les oscillations observées se situent dans une plage entre 40 et 60 Hz [Llinás 93]. Ces résultats ont été confirmés par beaucoup d'autres études durant les années 90 [Engel 91a], [Engel 91b], [König 91], [Schillen 91], [Löwel 92]. L'hypothèse proposée considère que la mise en synchronie de l'activité de neurones spatialement séparés correspond au liage des différentes propriétés qui caractérise la perception d'un même objet ou d'une partie de l'environnement [Singer 93]. Utiliser un liage temporel pour rassembler l'ensemble des traits qui constituent une unité perceptive associé à une activation impulsionnelle (potentiel d'action) permet également de superposer plusieurs représentations dans une même période pour peu que chaque liage ait lieu sur une phase différente. Ce nombre maximum de liage est cependant limité et dépend de la fréquence des oscillations [König 95].

Plusieurs modèles ont été proposés pour expliquer ce phénomène[Grossberg 91a]. Grossberg a également proposé que les mécanismes de mise en synchronie soient à l'origine des capacités de segmentation du système visuel [Grossberg 91b].

Attention visuelle et perception active

Notre hypothèse de mise en synchronie est présentée dans la section II.6.1. Si le monde ne contenait qu'un seul objet, alors l'activité de toutes les unités de traitement serait relative à cet objet. Nous considérons qu'il est suffisant pour construire un liage perceptif que le système n'observe qu'un objet à la fois. De cette façon, toutes les propriétés observées seraient relatives à cet objet.

Nous avons aussi conclu dans la section II.7.2 que notre système devait disposer de détecteurs de formes indépendants de la notion de position. Une condition suffisante consiste à disposer de détecteurs de forme situés pour des raisons de symétrie au centre du capteur visuel. Notre système doit donc explorer la scène pour en construire une représentation. La notion élémentaire de lieu est codée sous la forme de l'action effectuée pour placer un objet au centre de l'image, la forme est codée par des unités spécialisées dans l'extraction de motifs présents au centre du système de vision.

Un problème se pose alors : si le système ne peut déterminer la nature des objets en périphérie de son système visuel, sur quelle base s'effectue cette exploration visuelle ? Nous proposons une carte de saillance n'utilisant que des informations de bas niveau (contraste, couleur, mouvement, etc.). Cette carte de saillance peut être calculée à la fois en utilisant la notion d'information au sens de Shannon (la propriété la plus rare et la plus stable est celle qui porte le plus d'information) et un processus de compétition dynamique entre les centres d'intérêt pour éviter les minima locaux.

Les conditions relatives à la mise en synchronie semblent compatibles avec les résultats des expériences psychophysiques menés par Kevin O'Regan et Ronald Rensink à la fin des années 90 [O'Regan 96], [O'Regan 97]. Le principe de leurs expériences consiste à tester la capacité à détecter des changements dans une scène visuelle en utilisant divers types de distracteurs visuels. Les sujets observent successivement une image originale et une image retouchée contenant un changement important (jusqu'à un cinquième de la taille de l'image). Ce changement peut être un objet déplacé, un changement de couleur, un objet enlevé, etc. Pour que la variation entre les deux images n'attire pas l'attention, ils font clignoter des formes géométriques simples (mud-splash) au moment du passage de l'image originale à l'image retouchée. Les résultats de cette expérience permettent de mieux comprendre le fonctionnement du système visuel. Les résultats observés sont particulièrement surprenants. Si le sujet de l'expérience ne porte pas explicitement son attention sur la partie variable de l'image, alors il est totalement aveugle au changement. Il arrive que des sujets passent plus d'une minute pour trouver ce qui a changé entre les deux images. Ils ont également enregistré la direction du regard au cours de l'expérience, et ont montré que même si les sujets observaient un détail à seulement 1° d'angle visuel de la partie variable, ils étaient incapables de le détecter, et n'avait aucune conscience du changement. Cette expérience montre que l'homme ne voit précisément que ce sur quoi il porte explicitement son attention. Toute autre information semble ajoutée ou reconstruite par le cerveau. Un peu comme le remplissage de la tache aveugle dont nous n'avons absolument pas conscience.

Emotions et représentations subjectives

Le système limbique (Fornix, Noyaux antérieurs du thalamus, amygdale, hippocampe) est en particulier impliqué dans les émotions, l'apprentissage et la mémoire. Il existe de nombreux points communs entre notre système de représentation subjective et l'ensemble des composantes du système limbique [Rolls. 01]. En particulier le fait d'associer une émotion à toute perception.

II.9.5 Conclusion

L'analyse des propriétés nécessaires ou parfois suffisantes dont doit disposer un agent hédoniste situé et incarné, sans connaissances *a priori* sur son environnement pour accroître ses connaissances et ses fonctionnalités nous a permis en particulier d'exhiber un système ouvert de représentation et de fonctionnalité qui permette de développer un tel agent. Ce dernier apprend à généraliser les causes de ses variations d'états subjectifs. Pour les généraliser, il doit représenter le monde sous une forme totalement distribuée. Les représentations minimales peuvent être considérées comme équivalentes : nous les avons appelé unités de traitement. Nous avons vu que ces unités de traitement doivent s'observer, et qu'elles doivent prendre part dans un réseau. Nous avons aussi montré qu'elles doivent posséder plusieurs propriétés : la fidélité (les mêmes causes doivent produire les mêmes effets), la continuité (deux causes voisines doivent produire deux effets voisins), la comparabilité (deux effets doivent être comparables pour résoudre les compétitions), la superposabilité (le canal de communication doit permettre de faire transiter simultanément plusieurs messages ou encore les unités de traitement doivent pouvoir être simultanément dans plusieurs états d'activations) et la persistance (les unités de traitement doivent être actives pendant une certaine durée).

L'agent doit apprendre à se représenter l'environnement en le décomposant en catégories indépendantes et pertinentes. Sa perception est spectrale[10], il doit percevoir simultanément l'ensemble des catégories relatives aux propriétés du monde pour être en mesure de trouver la cause des variations de son état subjectif. L'agent a accès au monde à travers des capteurs et doit chercher à stabiliser sa perception. Son apprentissage doit être basé sur une exploration et une expérimentation de l'environnement. Le niveau de compréhension de l'agent réside dans sa capacité à trouver des catégories abstraites et de haut niveau. En cela, il aura la possibilité d'apprendre des fonctionnalités de plus en plus complexes. Plus il sera en mesure de construire des catégories complexes et abstraites du monde et de trouver de la causalité entre elles, plus les fonctionnalités qu'il développera seront d'un niveau élevé.

L'agent n'est pas développé pour résoudre un problème. Il est hédoniste et n'a qu'un objectif : accroître sa perception d'effets subjectifs. Pour cela il utilise un système de représentation ouvert. Ce type d'approche n'est pas spécifique à un type de problème précis, elle est générale quelque soit le problème.

Les conditions nécessaires ou suffisantes que nous avons exhibées dans ce chapitre sont relativement précises, pour autant elles ne permettent pas de construire un modèle unique. Le

[10] Spectrale dans le sens où elle agit de la même manière qu'un prisme sur de la lumière blanche.

chapitre suivant va s'attacher à préciser un modèle particulier.

Généralisation :	L'agent doit extraire l'information relative aux causes de la variation de ses perceptions subjectives.
Représentations distribuées :	Pour pouvoir généraliser, le système de représentations objectives est constitué d'unités de traitement distribuées
Unité de traitement :	Chaque unité de traitement se différencie au cours de l'évolution du système pour extraire une propriété particulière du contenu qu'elle observe. Chaque fois que la cause de son activation se trouve présente dans son espace d'observation, elle change d'état. L'état d'une unité de traitement traduit le résultat d'un test sur son espace d'observation. Les propriétés nécessaires de cet état d'activation sont précisées dans le Tableau II.3.
Champs récepteurs :	Chaque unité de traitement observe d'autres unités de traitement. Le champ récepteur d'une unité est l'ensemble des unités qu'elle observe.
Compétition :	Chaque champ récepteur doit être observé par plusieurs unités de traitement en compétition. Ces unités en compétition doivent apprendre à décomposer les motifs formés par l'état des unités du champ récepteur qu'elles observent dans une base. Cet apprentissage doit se faire à travers une compétition pour que deux unités ne se spécialisent pas dans l'extraction du même type de propriété. Au cours du processus de compétition, les unités doivent se différencier fonctionnellement.
Représentations hiérarchiques :	Il doit exister dans le système de représentations objectives des unités de traitement qui observent d'autres unités de traitement. Ces unités des traitement sont donc organisées en graphe et s'observent les unes les autres.
Liage temporel :	Le système de représentation objective de l'agent code toutes les propriétés des parties du monde de façon distribuée. Pour cela l'activité des unités de traitement doit être périodique. Pour coder le fait que ces propriétés sont relatives à une même partie du monde, les assemblées d'unités doivent être synchrones. Les assemblées sont distinguées par leur phase.

TAB. II.2 – Résumé des propriétés nécessaires du système de représentations objectives.

Fidélité :	Chaque unité de traitement doit extraire une information stable de l'environnement, c'est à dire un invariant perceptif. Placée dans les mêmes conditions initiales, l'unité de traitement doit s'activer de la même façon.
Continuité :	L'information extraite doit être continue. L'activité des unité de traitement doit varier faiblement lorsque l'environnement varie faiblement.
Comparabilité :	L'état d'activation des unités de traitement doit être comparable pour permettre le processus de compétition durant la spécialisation.
Superposabilité :	Les unités de traitement doivent pouvoir superposer plusieurs niveaux d'activité afin de participer à plusieurs liages temporels.
Persistance :	La persistance correspond à la durée d'activation d'une unité.

TAB. II.3 – Résumé des propriétés nécessaires des unités de traitement.

Chapitre III

Synthèse d'un modèle

Comme nous l'avons vu dans le chapitre précédent, l'ensemble des propriétés nécessaires ou suffisantes exhibées ne conduisent pas à un modèle unique, mais à une classe de modèles. Peu importe la façon dont sont réalisées les unités de traitement, pourvu qu'elles possèdent les propriétés voulues. Nous n'allons pas étudier toutes les instances possibles d'un tel jeu de propriétés dans ce chapitre, mais plutôt essayer de construire un modèle particulier qui satisfasse les propriétés que nous cherchons à obtenir.

Nous avons montré au cours de notre analyse qu'il suffit que le système que nous souhaitons modéliser soit constitué d'unités de traitement et que l'endroit où elles se trouvent permette d'abstraire la notion de sémantique. Le système doit disposer d'un ensemble fini d'algorithmes qui doivent suffire à engendrer toutes les propriétés recherchées[1]. La construction du modèle peut être comparée à un jeu de légo. La brique élémentaire est ce que nous avons désigné comme l'unité de traitement dans le chapitre II. Nous devons en particulier faire en sorte que notre modèle de l'unité de traitement tienne compte des propriétés de fidélité, continuité, comparabilité, et persistance. Nous devons également modéliser le code utilisé par ces unités pour communiquer leur états d'activations successifs.

Ces briques de base en se combinant doivent engendrer des propriétés nouvelles. Il va donc falloir construire des structures qui proposent des propriétés de plus haut niveau, des réseaux de ces unités de traitement. Le réseau est le graphe qui définit les relations entre unités élémentaires observées et observantes.

Nous allons dans un premier temps proposer un modèle particulier d'unité de traitement et dans un second temps nous modéliserons des propriétés de groupe. Modéliser c'est faire

[1] De la même façon que les tables d'addition, soustraction, multiplication et division ne portent que sur des chiffres et leur voisinage (pas des nombres), et sont en nombre fini.

des choix ; il faudra prendre garde que les décisions prises permettent toujours d'obtenir les propriétés que nous souhaitons.

Le modèle que nous allons construire est destiné à être implémenté sur plusieurs ordinateurs. Pour cela nous devons orienter nos choix de telle façon qu'ils occasionnent le moins de calculs possible.

III.1 Réseaux de neurones formels

Les unités de traitement nécessaires à un tel système font inévitablement penser à des réseaux de neurones formels. Pour autant il n'existe aucun modèle de neurones artificiels qui possèdent l'ensemble des propriétés que nous recherchons. S. Haykin présente un très grand nombre de modèles, ainsi que la démonstration de leur convergence dans son ouvrage *Neural Networks, A Comprehensive Foundation* [Haykin 94], qui est aujourd'hui une référence en la matière.

La classe d'agent que nous étudions est située par hypothèse, elle doit apprendre en ligne. Nous écartons donc toute approche qui utiliserait une phase d'apprentissage en utilisant une base d'exemple et une phase d'utilisation.

Nous écartons également les méthodes à apprentissage supervisé car l'évolution du système doit être ouverte. Il n'est pas concevable d'imaginer des couples d'entrées/sorties souhaitées et d'utiliser un algorithme de rétropropagation de gradient [Rosenblatt 58], [Rosenblatt 60] [Rosenblatt 62]. Nous utiliserons une méthode non supervisée de catégorisation.

L'ensemble fini d'algorithmes que nous allons construire doit nécessairement être local. Ces algorithmes ne doivent porter que sur des unités de traitement voisines[2]. Si ces algorithmes portaient sur la globalité des données, ils seraient spécifiques à un réseau particulier d'unités de traitement et ne seraient pas généraux[3].

Les approches non-supervisées semblent plus adaptées au modèle que nous souhaitons construire. Oja a introduit une méthode d'analyse en composantes principales (PCA) en 1982 [E. Oja 82]. L'approche de Kohonen et plus particulièrement la théorie des cartes auto-organisatrices (SOM[4]) [Kohonen 95] semble constituer un point de départ adapté. Cette approche conduit à la catégorisation d'un espace d'entrée. Après la convergence d'un tel réseau, des neurones voisins catégoriseront des propriétés voisines de l'espace. Les SOM nécessitent l'utilisation d'une distance entre les poids synaptiques et les données d'entrée. Il est également possible avec l'approche SOM de catégoriser et de suivre les relations topologiques de catégories qui évoluent au cours du temps [Paquier 98] [Bouchired 98]. Malheureusement l'algorithme de Kohonen utilise une propriété globale pour déterminer quel neurone est le plus proche d'une catégorie présentée en entrée. Ce procédé appelé "winner take all" est en contradiction avec le fait que les algorithmes ne doivent pas porter sur la totalité du système. De plus, ce modèle

[2] Voisines au sens du codage par position

[3] Pour poursuivre le parallèle avec le codage par position, les règles d'arithmétique ne portent pas sur les nombre mais sur les chiffres et leur voisinage, par exemple la prise en compte d'une retenue dans la règle d'addition

[4] SOM : Self-Organizing Maps

n'est pas très adapté aux réseaux multicouches.

Le système de représentation objective est un réseau d'unité de traitement. Ce réseau doit être suffisamment profond pour permettre de construire des catégories complexes et de plus en plus indépendantes des données brutes que l'agent observe.

Les modèles d'inspiration biologique semblent plus indiqués pour développer notre modèle. La majorité est inspirées du modèle "intègre et tire" de Lapicque [Lapicque 07]. Dans les cas les plus simples, chaque synapse est modélisée par un nombre réel. Le neurone est modélisé par un potentiel qui est intégré au cours du temps, lorsque ce potentiel dépasse un seuil le neurone génère un potentiel d'action. Certains modèles « intègre et tire » sont beaucoup trop riches et complexes pour être mis en oeuvre de façon efficace sur les ordinateurs dont nous disposons actuellement, en particulier ceux basés sur les méthodes de Hodgkin-Huxley [Hodgkin 52]. Plusieurs heures sont parfois nécessaires pour simuler avec beaucoup de précision l'évolution de l'ensemble des paramètres qui caractérisent l'activité d'un neurone pendant une seule seconde. Parmi les simulateurs les plus connus on peut citer Neuron développé par M.L Hines [Hines 89], [Hines 01] et Genesis développé par M. A. Wilson [Wilson 89].

L'algorithme local le plus adapté à notre modèle est celui proprosé par D. Hebb en 1949 [Hebb 49]. Il ne prend en compte qu'un couple de neurones pré et post synaptique : il respecte donc parfaitement la condition de traitement local qu'impose un codage par position. Il permet de faire évoluer le poids synaptique en fonction de l'état de ces deux neurones. Cet algorithme à été récemment étudié de façon exhaustive par Gerstner [Gerstner 99].

Plus récemment, Olshausen [Olshausen 00] et Bell [Bell 97] ont montré qu'il était possible de générer des bases de représentation des images naturelles en utilisant un grand nombre de ces images. Ils ont en particulier montré que ces bases pouvaient être « sparce », c'est à dire creuses. Ces méthodes sont statistiques et n'utilisent pas de neurones formels. Elles montrent néanmoins qu'il est possible de décomposer des images de façon distribuée et de construire ces bases en utilisant seulement des images naturelles.

L'aspect impulsionnel du neurone biologique est à l'origine de modèles de codage temporel très efficaces, en particulier le codage par rang [Gautrais 98]. Ce type de codage a conduit au développement de SpikeNET qui est un simulateur de réseaux de neurones impulsionnel et asynchrone [Delorme 99], [Delorme 03]. Ce simulateur utilise un modèle intègre et tire, des neurones à seuil, et une fuite. Il utilise un modèle de propagation dit « event driven » qui permet d'économiser beaucoup de temps de calcul. Nous utiliserons l'aspect « event driven ». Cette approche est rendue possible par l'aspect impulsionnel de la décharge des neurones dans ce modèle. Le fait qu'il puisse être inactif permet d'économiser énormément de calculs. Ce simulateur a permis de montrer qu'il était possible de reconnaître des visages, du mouvement, des caractères d'imprimerie, etc (Voir [Thorpe 98], [Thorpe 00], [Paquier 00a], [Paquier 00b]). Il ne permet pas d'extraire des catégories de façon non supervisée. Nous retiendrons de cette approche l'aspect « event driven » et l'aspect impulsionnel hérité de l'approche intègre et tire.

Il n'existe aucun modèle de réseaux formels qui réponde à l'ensemble de nos contraintes. Nous allons construire un modèle en combinant partiellement des sous-parties des modèles précédents tout en ajoutant des propriétés nouvelles.

III.2 Entrées et codage par position

Notre analyse nous a montré qu'un codage par position associé à un nombre fini d'états par position était suffisant pour qu'il existe un agent suffisamment adapté, c'est à dire comportant suffisamment d'unités de traitement pour représenter un ensemble de catégories. Pour cela, nous devons créer une interface entre le monde physique et le système en traduisant les propriétés du monde dans un codage par position. Dans le modèle, toutes les entrées du système sont des ensembles d'unités spatialement organisées ayant à la fois les propriétés des unités de traitement (sortie) et réalisant des mesures dans le monde physique (entrée). Nous appellerons ces unités d'interface des « convertisseurs ». L'état de chaque convertisseur est relatif à la présence, ou non, d'une propriété particulière du monde.

Les entrées du système sont donc des ensembles de convertisseurs spatialement organisés, c'est à dire des images reflétant l'état des convertisseurs[5]. Pour l'observation visuelle, il doit utiliser des images de luminance, ou de couleur. Pour l'observation auditive il doit utiliser des images de fréquences, etc. Dès cette étape le sens est abstrait dans la position. C'est la position relative des convertisseurs et leur état d'activité (ensemble d'états finis) qui va permettre aux unités qui les observent d'extraire des catégories en fonction de leur position.

III.3 Temps et causalité

Notre analyse nous a conduit à considérer que la causalité des événements devait jouer un rôle central pour ce type de système. Le fait qu'un événement se passe avant un autre véhicule de l'information nécessaire au fonctionnement des unités de traitement. La façon dont nous allons modéliser le temps est donc centrale pour le modèle que nous devons réaliser.

Nous choisissons un modèle à temps discret et sans mémoire. Si nous appelons $\mathscr{M}$ l'ensemble des paramètres du modèle global, nous aurons $\mathscr{M}(t+1) = F(\mathscr{M}(t))$.

Définissons quelques fonctions mathématiques utiles pour la construction de notre modèle. Nous considérons une fonction d'attraction linéaire L_α^+ définie par :

$$\forall x,y \in \mathbb{R}, \forall \alpha \in [0,1], L_\alpha^+(x,y) = (1-\alpha)x + \alpha y \tag{III.1}$$

et une fonction de répulsion linéaire L_α^- définie par :

$$\forall x,y \in \mathbb{R}, \forall \alpha \in [0,1], L_\alpha^-(x,y) = (1+\alpha)x - \alpha y \tag{III.2}$$

Notons que L_α^+ est une fonction convexe. Nous allons essayer de construire un modèle qui soit le plus simple possible pour les propriétés nécessaires dont nous cherchons à rendre

[5]Il n'est pas possible de créer une interface entre un nombre réel et une unité de traitement. Un nombre ne contient pas une information rassemblée dans un symbole unique. En observant ce nombre il n'est pas possible de généraliser une propriété. Pour cela il faut pouvoir observer plusieurs aspects de ce nombre et trouver quels aspects comptent. Par exemple 2 est 4/2, plus petit que 10, positif, etc.

compte. Nous faisons donc l'hypothèse de variation linéaire des paramètres entre chaque instant. Nous introduirons les différents paramètres qui constituent une unité de traitement progressivement.

III.4 L'unité de traitement

Intéressons nous dans un premier temps aux unités de traitement. Nous savons que les unités de traitements s'observent les unes les autres et qu'aucune n'observe la totalité. Les unités de traitement sont elles-mêmes des agents autonomes organisés en réseau. Les unités de traitement sont au carrefour de deux flux d'information : le flux d'information objective et le flux d'information subjective (voir Figure III.1. Il doit exister un canal de communication entre chaque unité observante et observée.

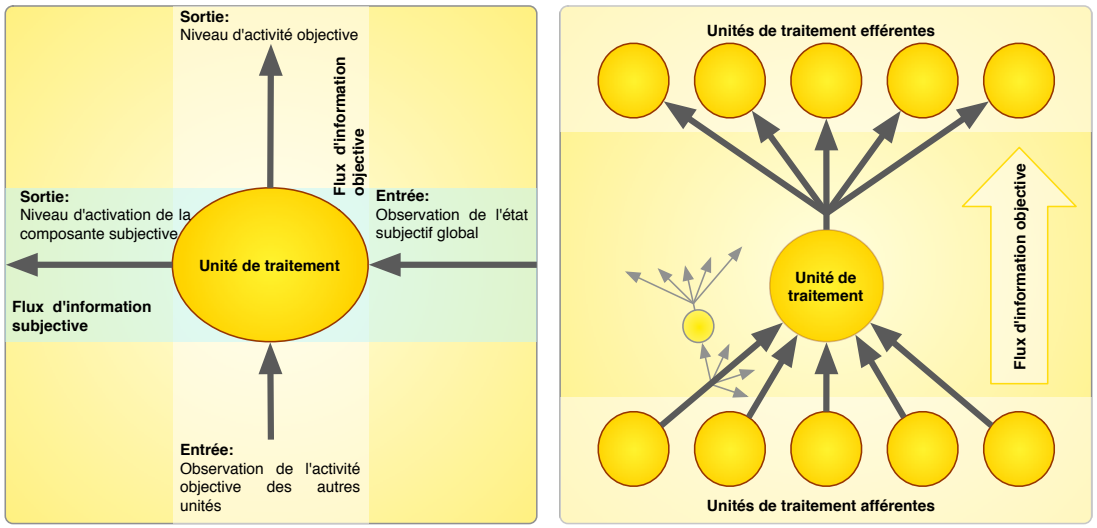


FIG. III.1 – Entrées et sorties d'une unité de traitement.

Le flux d'information objective est relatif à la catégorisation et la création des représentations distribuées. Chaque unité possède un ensemble d'entrées et une seule sortie qu'elle réplique pour chaque unité qui l'observe. Le flux d'information subjective est relatif à l'apprentissage de nouveaux états récompense, et contribue à l'état subjectif global du système. Chaque unité contribue, quand elle s'active, à l'état subjectif global, et utilise ce dernier pour adapter sa propre composante subjective. Nous allons dans un premier temps modéliser le flux objectif, puis le flux subjectif et enfin nous modéliserons l'interaction de ces deux flux.

III.5 Flux d'information objective : Processus de catégorisation

Une unité de traitement U est caractérisée par des entrées et une sortie. Les entrées de l'unité de traitement U sont constituées par l'ensemble des activités objectives des unités qu'elle observe (voir Figure III.1). Nous appellerons unités afférentes de U l'ensemble des unités observées. U possède également une activité objective qui est observable par d'autres unités de traitement. A chaque instant t une unité de traitement doit réaliser un calcul sur les données qu'elle observe. Le résultat de ce calcul est interprétable comme le fait que l'ensemble des données d'entrée appartienne à une catégorie ou pas.

III.5.1 Le zéro

Tout codage par position ne vaut que s'il est possible de coder l'absence d'une chose en utilisant un signe particulier. Ce signe doit signifier le vide, l'absence d'une propriété et porte également une information. Dans notre modèle cela signifie qu'un état particulier de l'unité de traitement doit véhiculer ce sens. Nous tiendrons compte de cette propriété dans la construction du modèle. Une unité de traitement devra pouvoir être inactive et porter une information au reste du système à travers cette inactivité.

III.5.2 Modèle d'appartenance à une catégorisation

Le processus de catégorisation doit respecter au moins les deux contraintes suivantes :

– Initialement le système ne sait rien sur son environnement. La catégorie extraite par une unité évolue donc au cours du temps et il faut donc modéliser ce processus d'adaptation.

– Chaque unité de traitement doit posséder la propriété de fidélité quand elle a appris, c'est à dire que pour des entrées identiques, elle doit avoir une sortie identique. Le processus d'adaptation doit converger vers l'extraction d'une catégorie fixe, c'est à dire que les conditions d'appartenance à une catégorie doivent converger vers un ensemble fixe.

Nous allons construire le modèle de façon progressive. Nous allons dans un premier temps modéliser la propriété de mesure d'appartenance à une catégorie.

Notons $A_i(t)$ l'activité d'une unité de traitement, c'est-à-dire l'ensemble des informations qui la caractérise à chaque instant. $A_i(t)$ est un ensemble de paramètres qui varient à chaque instant. L'activité doit posséder au moins deux propriétés : La première consiste à informer que l'état des entrées appartient à une catégorie ou non. Cette propriété peut être modélisée par un paramètre booléen. Nous utiliserons plutôt une logique à 3 états que nous détaillerons plus loin. La seconde est relative à la propriété de comparabilité exhibée dans l'analyse. Les unités en compétition doivent se spécialiser dans l'extraction de motifs particuliers. Pour cela, l'activité $A_i(t)$ doit permettre de comparer l'état des unité en compétition. La comparabilité se traduit par une relation d'ordre dans le modèle. Dire qu'une unité U_i gagne une compétition à l'instant t face à U_j revient à ce que l'activité $A_i(t)$ soit supérieure au sens de la relation d'ordre de comparabilité à $A_j(t)$.

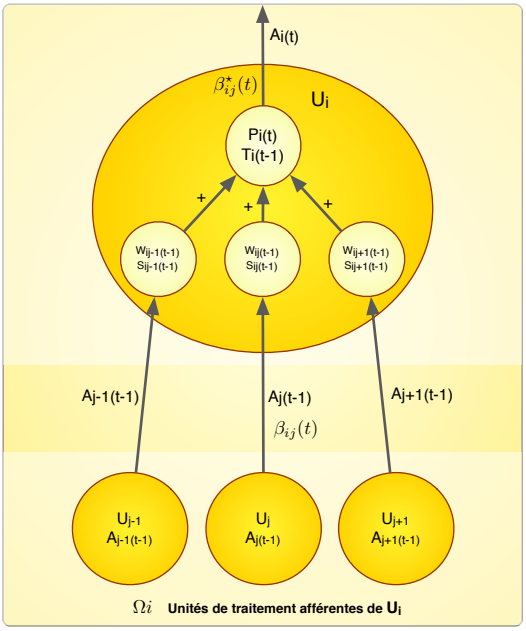


FIG. III.2 – Intégration, calcul de ressemblance.

L'activité sera donc au moins composée d'un paramètre destiné à calculer l'appartenance des entrées d'une unité à la catégorie qu'elle code. Nous noterons ce paramètre d'entrée $\beta_i(t)$ avec $\beta_i(t) \in [-1,1]$ (voir Figure III.2). La valeur de $\beta_i(t)$ traduit une logique à 3 états : si $\beta_i(t) \in]0,1]$ alors le motif appartient à la catégorie codée par U_i, si $\beta_i(t) = 0$ alors le motif est neutre vis-à-vis de U_i, et si $\beta_i(t) \in [-1,0[$ alors le motif appartient à l'opposé de la catégorie codée par U_i.

Le processus d'intégration est basé sur un modèle « event driven » très simple. Chaque connexion entre unités de traitement est ramenée à un poids synaptique $W_{ij}(t)$ et un état présynaptique $S_{ij}(t)$. S_i et W_j sont leur vecteurs respectifs. Les unités de traitement doivent être indépendantes et n'ont accès à l'état des unités qu'elle observe qu'à travers les messages qu'elles reçoivent. L'état présynaptique S_i est un paramètre utilisé pour estimer l'état d'activation des unités observées. Initialement, les états présynaptiques sont nuls. Les poids synaptiques sont choisis aléatoirement dans l'intervalle $[0,1]$. Ces paramètres sont présentés figure III.2.

L'intégration est réalisée à l'aide d'un potentiel P_i et d'un seuil T_i. Chaque unité de traitement reçoit la composante $\beta_i(t)$ de chacune de ses unités afférentes. L'état d'une unité peut être nul, dans ce cas, $\beta_i(t) = 0$.

Nous appellerons Ω_i l'ensemble des afférents de l'unité U_i et $\Omega_i^+(t)$ le sous-ensemble de Ω_i pour lequel les $\beta_i(t)$ correspondants sont strictement positifs à un instant donné. C'est sur

ce sous-ensemble d'afférents que porte notre calcul d'intégration.

Le motif observé par l'unité de traitement est constitué des $\beta_i(t) \in \Omega_i^+(t)$. La catégorie codée par l'unité est fonction du vecteur de poids $W_i(t)$ et du seuil T_i. Le calcul du potentiel P_i est une première étape du calcul d'appartenance du motif d'entrée à la catégorie codée par l'unité.

$$P_i(t+1) = L_{\alpha_p}^-\left(P_i(t), \sum_{U_j \in \Omega_i^+(t)} W_{ij}(t)\right) \tag{III.3}$$

Le potentiel est calculé comme une somme pondérée entre la valeur précédente du potentiel et la somme des poids synaptiques. Cette pondération correspond à une fuite exponentielle de premier ordre que subit le potentiel entre chaque pas de temps de telle sorte que si une unité ne reçoit plus d'entrées alors elle verra son potentiel tendre vers 0.

Quand le potentiel est suffisamment élevé pour dépasser le seuil alors l'unité génère une nouvelle valeur appelée $\beta_i^\star(t)$ qui correspond au niveau de ressemblance du motif observé avec la catégorie codée par l'unité U_i.

Définissons une norme que nous allons utiliser par la suite.

Pour $x \in \mathbb{R}^n$ nous définissons respectivement x^+ et x^- la somme des composantes positives et négatives de x.

$$x^+ = \sum_{x_i>0} x_i \text{ et } x^- = \sum_{x_i<0} x_i \tag{III.4}$$

$(\cdot)^+$ est une norme si et seulement si $x^+ = -x^-$ c'est à dire si la somme des composantes de x est nulle. Nous avons alors :

$$x^+ = 0 \Leftrightarrow x = 0 \tag{III.5}$$

$$(x+y)^+ \leqslant x^+ + y^+ \tag{III.6}$$

$$(\lambda x)^+ = |\lambda| x^+ \tag{III.7}$$

Pour une lente évolution de $W(t)$ au cours du temps (*i.e.* pour $\alpha_W \ll 1$, α_W correspond au coefficient d'apprentissage de l'unité et sera défini plus loin) et en vertu de III.3, $P_i(t)$ est borné : $W_i^-(t) \lesssim P_i(t) \lesssim W_i^+(t)$. Le cas de l'égalité positive correspondant à l'observation systématique du motif correspondant le plus à la catégorie à extraire, et le cas négatif à l'observation d'un motif correspondant à l'opposé de la catégorie à extraire.

Le processus de catégorisation est basé sur une fonction de transfert linéaire par morceaux. Elle exprime l'appartenance du motif formé par l'activité positive des unités $U_j \in \Omega_i^+(t)$ (voir Figure III.3, droite). Cette valeur d'appartenance est appelée $\beta_i^\star(t)$. C'est un résultat intermédiaire qui n'est pas accessible par les autres unités de traitement. La fonction de transfert dépend de $P_i(t)$ et $T_i(t)$ et est donnée par la relation suivante :

$$\beta_i^\star(t) = \begin{cases} \frac{P_i(t)-T_i(t)}{1-T_i(t)} & \text{si } P_i(t) > T_i(t) \\ \frac{P_i(t)+T_i(t)}{1-T_i(t)} & \text{si } P_i(t) < -T_i(t) \\ 0 & \text{sinon} \end{cases} \tag{III.8}$$

Pour simplifier cette relation, il est possible d'exprimer $P_i(t)$ et $T_i(t)$ en fonction de $W_i^+(t)$. Le seuil $T_i(t)$ de U_i peut être exprimé en fonction de $W_i^+(t)$:

$$T_i(t) = \gamma_i(t) W_i^+(t) \tag{III.9}$$

De même $P_i(t)$ peut aussi être exprimé en fonction de $W_i^+(t)$

$$P_i(t) = \phi_i(t) W_i^+(t) \tag{III.10}$$

De cette façon, la fonction de transfert se simplifie par W_i^+ (voir Figure III.3). Ce qui donne :

$$\beta_i^\star(t) = \begin{cases} \frac{\phi_i(t)-\gamma_i(t)}{1-\gamma_i(t)} & \text{si } P_i(t) > T_i(t) \\ \frac{\phi_i(t)+\gamma_i(t)}{1-\gamma_i(t)} & \text{si } P_i(t) < -T_i(t) \\ 0 & \text{sinon} \end{cases} \tag{III.11}$$

Puisque à chaque instant $P_i(t) \leqslant W_i^+(t)$ alors $\phi_i(t) \in [-1,1]$. De plus $T_i(t) \leqslant W_i^+(t)$ ce qui conduit à $\gamma_i(t) \in [-1,1]$. Le résultat de la fonction de transfert vérifie donc $\beta_i^\star(t) \in [-1,1]$.

La fonction de transfert sépare linéairement l'ensemble des entrées possibles (Voir Figure III.3, droite). C'est une propriété aussi ancienne que les premières études de McCulloch et Pitts sur les réseaux de neurones formels [McCulloch 43]. On comprend de fait qu'un XOR ne peut pas être réalisé en moins de deux couches. La ressemblance s'arrête là, la façon dont les poids vont s'adapter est ici non supervisée et basée en partie sur la règle de Hebb [Hebb 49]. Si l'espace d'entrée est de dimension n ($Card(\Omega_i) = n$), alors la fonction de transfert sépare l'espace en 3 régions à l'aide de 2 hyperplans parallèles : la partie positive qui correspond à l'appartenance à une catégorie, la partie centrale correspond à la neutralité de U_i relativement à une catégorie et la partie négative correspond à l'opposé sémantique de la catégorie. Le fait que la sortie d'une unité de traitement puisse être négative sert uniquement pour l'adaptation des poids (voir règle d'apprentissage plus loin), pas pour l'intégration qui est effectuée sur $\Omega_i^+(t)$. Nous discuterons la notion d'opposé sémantique dans les perspectives.

La partie droite de la figure III.3 illustre un cas particulier où le nombre d'afférents est égal à 2. Dans ce cas la partie positive de la fonction de transfert est une droite qui délimite la région correspondant à la catégorie à extraire.

III.5.3 Compétition

Comme nous l'avons vu, des unités en complétion observent les mêmes données d'entrées et ne doivent pas extraire la même catégorie. Elles doivent donc échanger des informations pour

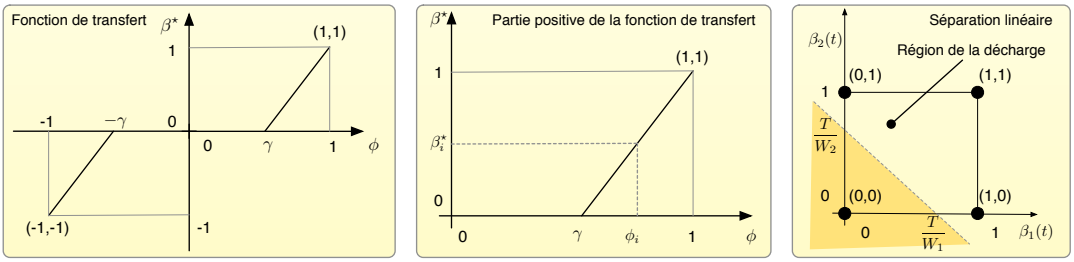


FIG. III.3 – Fonction de transfert et séparation linéaire.

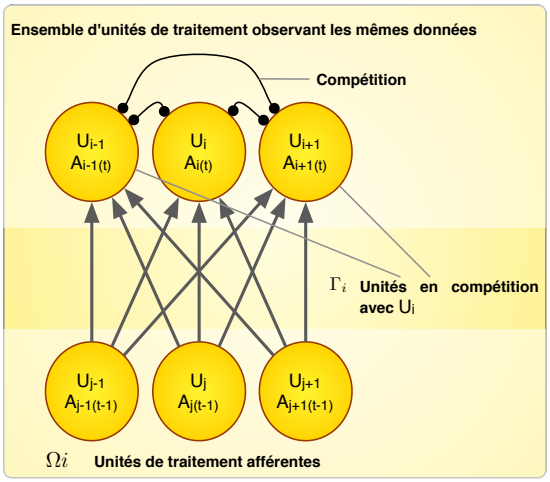


FIG. III.4 – Ensemble d'unité en compétition.

s'assurer qu'elles ne vont pas apprendre la même catégorie (voir Figures III.1, III.4). Seule l'unité qui aura localement gagné une compétition pourra s'adapter au motif qu'elle observe.

Le fait d'adopter un système de codage par position conduit à n'utiliser que des règles locales entre unités de traitement. Aucun calcul ne doit porter sur la globalité des unités de traitement. Les algorithmes que nous mettons en place ici ont le statut des règles arithmétiques dans le cas des systèmes de numération par position. C'est parce que ces algorithmes ne portent que sur des formes locales et ne se préoccupent pas du sens véhiculé par l'état de l'unité (ici le sous-réseau de compétition) que le système de représentation est ouvert. Notons Γ_i l'ensemble des compétiteurs de l'unité U_i.

La figure III.5 illustre la solution que nous avons choisie pour mettre en oeuvre une comparaison locale qui ne demande pas qu'un algorithme cherche globalement un gagnant, c'est le cas dans les cartes auto-organisatrices de Kohonen ([Kohonen 95]). Chaque unité de traitement doit savoir si elle a gagné une compétition ou non. Pour cela une unité de traitement observe

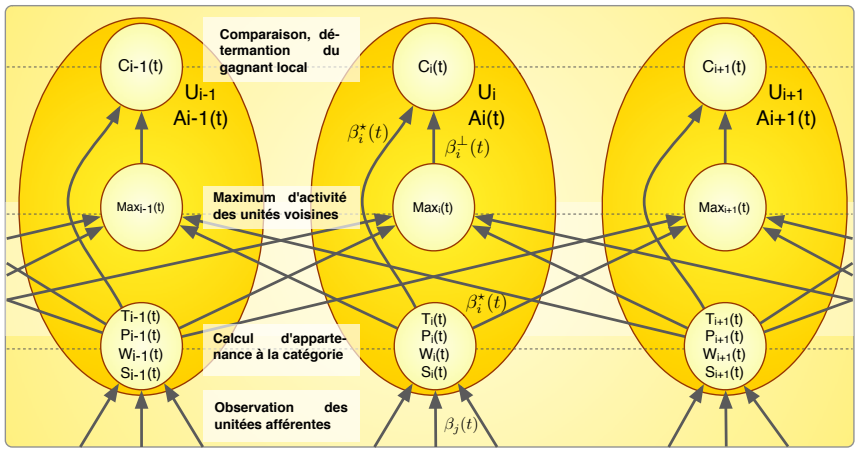


FIG. III.5 – Mécanisme de compétition locale.

l'ensemble des $\beta^\star(t)$ calculés par les unités avec lesquelles elle est en compétition. Nous appelons $\beta_i^\perp(t)$ le maximum des $\beta^\star(t)$ sur Γ_i. Cette valeur est comparée au propre $\beta^\star(t)$ de cette unité.

$$\beta_i^\perp(t+1) = \max_{U_j \in \Gamma_i} \beta_j^\star(t) \tag{III.12}$$

Pour pouvoir déterminer le maximum d'activité des unités voisines, il faut attendre un pas de temps supplémentaire. La comparaison de la valeur propre à l'unité avec le maximum des activités des unités en compétition doit s'opérer sur des valeurs issues du même pas de temps. L'étape d'intégration consomme un pas de temps, la comparaison aussi, la sortie du système d'identification de catégorie prend donc deux pas de temps.

Cette $\beta_i^\perp$ est aussi une des valeurs qui constitue l'activité de A_i, elle n'est toujours pas la valeur de sortie, car elle ne tient pas compte de la propriété de persistance.

III.5.4 Adaptation des poids et du seuil

Seules les unités actives et qui ont gagné une compétition sont susceptibles d'apprendre. Une autre règle relative au flux subjectif doit être également respectée. Il en sera question plus loin.

Les deux paramètres liés à la séparation des entrées sont les poids $W(t)$ et le seuil $T(t)$. La figure III.6 illustre un cas simple où l'espace d'entrée est de dimension 2. L'hyperplan coupe les axes en $\frac{T_i}{W_{ij}}$. L'équation de l'hyperplan est donc $\sum_{j \in Card(\Omega_i)} \frac{T_i}{W_{ij}}.x = 1$. On voit que plus T_i sera grand, plus l'unité sera sélective. Le contrôle que nous devons appliquer pour faire varier la sélectivité de l'unité consiste à modifier son seuil. Une unité qui ne s'active jamais est inutile au système, il faut donc que son seuil baisse pour accroître le nombre de motifs susceptibles de

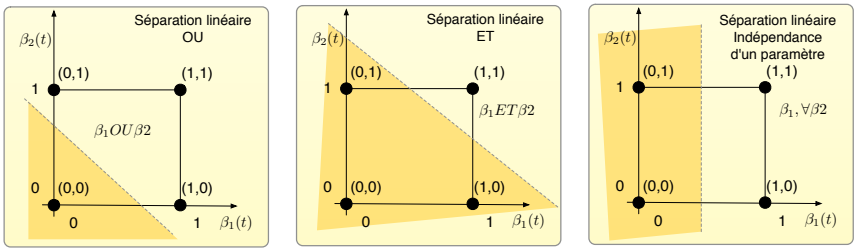


FIG. III.6 – U_i se comporte comme un ensemble de portes logiques adaptatives.

l'activer. De la même façon une unité qui s'active trop n'est plus informative pour le reste du système, il faut donc que son seuil croisse pour diminuer le nombre de motifs qui permettent de l'activer. De cette façon le critère est local à l'unité. Pour cela nous utilisons l'activité moyenne de l'unité. L'évolution du seuil T_i de U_i est donnée par :

$$\gamma_i(t+1) = \begin{cases} L^{+}_{\alpha_T^{up}}\left(\gamma_i(t), \gamma_i^l\right) & \text{si } E_i(t) > T_i^E \\ L^{-}_{\alpha_T^{down}}\left(\gamma_i(t), 0\right) & \text{si } P_i(t) > 0 \\ & \text{et } \beta_i^{\perp}(t) = 0 \\ \gamma_i(t) & \text{sinon} \end{cases} \tag{III.13}$$

Où γ_i^l modélise la sélectivité de l'unité U_i, c'est à dire la valeur maximale que le seuil T_i peut atteindre, E représente l'activité moyenne de la sortie et $\beta_i(t)$ sa sortie.

$$E_i(t+1) = L^{-}_{\alpha_E}\left(E_i(t), \beta_i(t)\right) \tag{III.14}$$

T_i^E représente le maximum d'activité moyenne avant que le seuil ne remonte. α_T^{down} représente le facteur d'attraction du seuil quand l'activité moyenne du neurone n'est pas suffisante, et α_T^{up} permet de remonter le seuil quand l'unité n'est plus suffisamment sélective. Plus le seuil d'une unité est haut, plus cette unité est sélective, c'est-à-dire qu'elle a appris une catégorie. Pour conserver cette propriété et ne pas remettre en cause le rôle de cette unité trop tôt, nous choisissons α_T^{up} plus grand que α_T^{down}.

Si une unité perd une compétition, alors son seuil ne descend pas. De ce fait, un groupe d'unité en compétition peut apprendre des motifs différents de façon séquentielle. Si aucune ne s'active pour un motif d'entrée toutes les unités baissent leur seuil. La première fois qu'une unité s'active pour cette configuration elle gagne la compétition et empêche les autres de descendre leur seuil, de fait elle les empêche également d'apprendre ce motif. Lorsque l'unité qui a gagné apprend, elle adapte ses poids et son seuil. La fois suivante, elle sera plus sélective au motif responsable de son apprentissage et aura moins de chance d'être activée par un motif voisin de celui qu'elle cherche à extraire. A ce moment là, il y aura à nouveau des cas où aucune ne s'activera. Toutes baisseront leurs seuils (sachant que le seuil de la première unité ayant appris sera beaucoup plus haut que celui des autres unités, car la remontée est plus forte que

la descente). Une unité différente de la première pourra alors s'adapter à un nouveau motif et ainsi de suite. Grâce à cette règle de descente et de montée du seuil, des unités en compétition sont capables d'apprendre des motifs présentés séquentiellement. Nous appelons cette propriété compétition temporelle. Une information qui pénètre dans le système empruntera un chemin différent en fonction du motif qui la caractérise (Voir Figure II.26).

L'apprentissage des poids est non supervisé [Ruf 98, Ruf 97, Senn 01]. Initialement, les poids W_{ij} de l'unité U_i sont distribués aléatoirement dans l'intervalle $[0,1]$. Si une unité s'active et qu'elle gagne une compétition locale, c'est à dire si $\beta_i^\star(t) > \beta_i^\perp(t)$ (cette opération est effectuée à l'étape C_i, figure III.5), alors elle s'adapte à la cause de sa victoire pour augmenter ses chances de gagner la fois suivante. Les poids sont attirés vers l'état présynaptique S_i de l'unité U_i [Hebb 49].

$$W_{ij}(t+1) = \begin{cases} L^-_{\alpha_i^w}\big(W_{ij}(t), S_i\big) & \text{si } \beta_i^\star(t) > \beta_i^\perp(t) \\ L^+_{\alpha_i^w}\big(W_{ij}(t), S_i\big) & \text{sinon} \end{cases} \tag{III.15}$$

Où l'état présynaptique S_i est calculé pour chaque afférent dans Ω_i

$$S_i(t+1) = L^-_{\alpha_S}\big(S_i(t), \beta_i(t)\big) \tag{III.16}$$

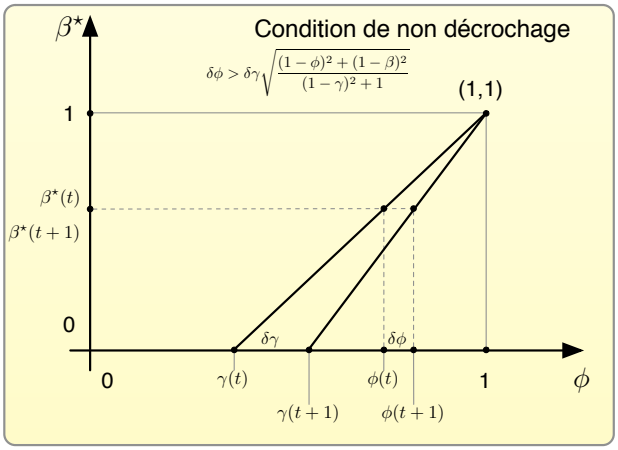


FIG. III.7 – Condition de non décrochage.

Plus une unité gagne de compétition plus elle s'adapte à la cause de son activité. Cette règle est inspirée de la règle de Hebb [Hebb 49]. Un poids synaptique augmente si l'unité post-synaptique est activée de façon causale par l'unité présynaptique et baisse si l'activité post-synaptique n'est pas due à l'activité présynaptique. Quand l'unité post-synaptique ne s'active pas, alors le seuil baisse ce qui a pour effet de relâcher la contrainte de sélectivité.

L'activité d'entrée peut être négative puisque nous comptabilisons ici la valeur des $\beta_i(t)$ reçus et qu'ils peuvent potentiellement être négatifs. Ceci permet à l'hyperplan de couvrir tous les cas possibles de séparation linéaire. Lorsqu'un poids W_{ij} tend vers 0, alors l'unité U_j observée n'influence plus l'activation de U_i. La loi d'activation de U_i sera indépendante de l'état de U_j (Voir figure III.6, droite).

Lorsque l'unité apprend la cause de son activation, elle doit en même temps respecter une condition dite de « non décrochage ». Supposons que l'unité observe exactement les mêmes données d'entrées, c'est à dire le vecteur d'entrée formé des β est le même entre deux pas de temps $\beta(t) = \beta(t+1)$. Lorsque le seuil remonte, la réponse de l'unité ($\beta^\star(t)$ peut être plus faible au second pas de temps si on ne respecte pas la condition de non décrochage. Cela peut conduire à une oscillation d'appartenance à une catégorie, car deux unités peuvent entrer dans un cycle ou un pas de temps sur deux. Dans ce cas elles vont gagner puis perdre une compétition, donc s'adapter aux mêmes causes. Pour que cela ne se produise jamais, il suffit que l'augmentation du seuil soit telle que :

$$\delta\phi > \delta\gamma\sqrt{\frac{(1-\phi)^2+(1-\beta)^2}{(1-\gamma)^2+1}} \tag{III.17}$$

Ce qui revient à :

$$\alpha^W > \alpha_T^{up}\frac{\gamma^l-\gamma}{Sj-Wj}\sqrt{\frac{(1-\phi)^2+(1-\beta)^2}{(1-\gamma)^2+1}} \tag{III.18}$$

La figure III.7 montre la valeur limite d'accroissement de γ avant que le décrochage ait lieu.

Plus une unité gagne des compétitions, plus elle devient sélective à une catégorie. Nous considérons qu'une unité atteint sa stabilité quand elle gagne toutes ses compétitions.

III.5.5 Persistance

L'unité de traitement doit être en mesure d'associer des événements quelconques. Pour cela, la propriété de persistance permet à une unité de continuer à véhiculer un message d'appartenance à une catégorie même si la cause a disparu. De cette façon une unité placée en amont dans le réseau pourra associer deux événements voisins dans le temps. C'est à la suite de cette étape que la sortie de l'unité est calculée ($\beta_i(t)$). Pour cela nous modélisons un compteur. La figure III.8 illustre la position de ce compteur dans l'unité.

$\Pi_i(t)$ est un compteur qui s'incrémente chaque fois que le système gagne une compétition et se réinitialise au bout de Π^m pas de temps.

$$\Pi_i(t+1) = \begin{cases} \Pi_i(t)+1 & \text{si } \Pi_i(t) < \Pi_i^m \\ & \text{et } \beta_i^\star(t) > 0 \\ 0 & \text{sinon} \end{cases} \tag{III.19}$$

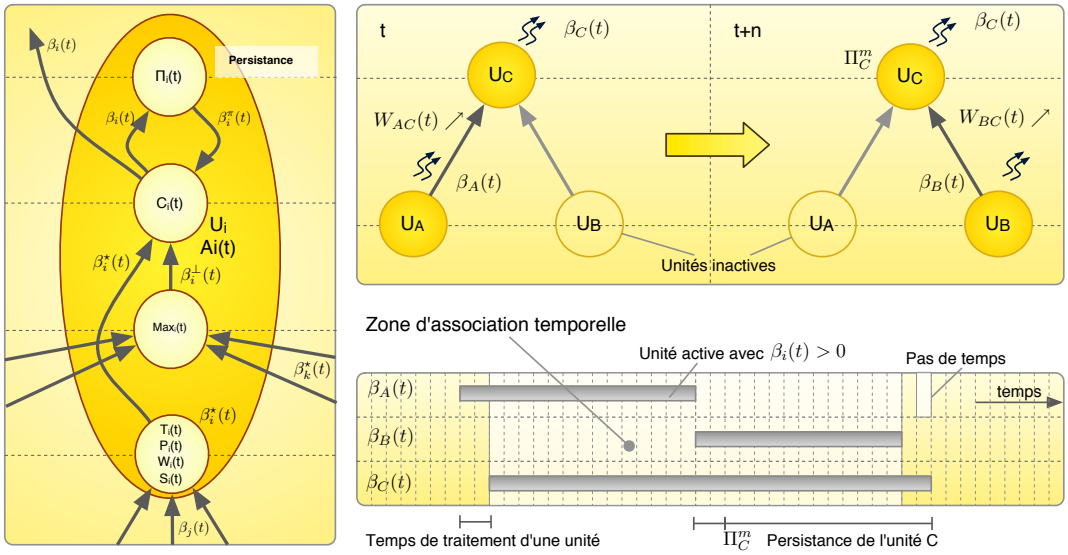


FIG. III.8 – Propriété de persistance et associations temporelles.

Tant que le décompte n'est pas terminé, l'unité reste active et fournit le maximum calculé par $\beta_i^{\Pi}(t)$ comme sortie effective de l'unité. De cette façon l'unité reste active pendant au moins Π^m pas de temps alors que la cause de cette activité a disparu.

$$\beta_i^{\Pi}(t+1) = \begin{cases} \max(\beta_i^{\Pi}(t), \beta_i^{\star}(t+1)) & \text{si } \Pi_i(t) > 0 \\ 0 & \text{sinon} \end{cases} \tag{III.20}$$

$$\beta_i(t+1) = \begin{cases} \max(\beta_i^{\Pi}(t), \beta_i^{\star}(t)) & \text{si } \beta_i^{\star}(t) > \beta_i^{\perp}(t) \\ & \text{ou } \beta_i^{\star}(t) < 0 \\ 0 & \text{sinon} \end{cases} \tag{III.21}$$

Lorsque le compteur est réinitialisé, $\beta_i^{\Pi}(t)$ est réinitialisé à 0. La sortie $\beta_i(t)$ est la plus grande valeur qu'a atteint la sortie intermédiaire depuis Π^m pas de temps, c'est à dire celle qui a gagné une compétition ($\beta_i^{\star}(t) > \beta_i^{\perp}(t)$). Si $\beta_i^{\star}(t)$ est négative, alors elle est simplement recopiée en sortie et n'interagit pas avec le système de persistance.

En pratique cette propriété doit permettre d'associer des unités quel que soit leur lieu, donc quel que soit leur sens, sur la base de propriétés temporelles. La figure III.8 illustre la façon dont une unité peut apprendre à associer deux informations en utilisant la propriété de persistance. On notera qu'une unité de traitement donne une réponse après deux pas de temps de traitement de l'information. Durant le premier pas de temps l'unité détermine si les données qu'elle observe appartiennent bien à la catégorie pour laquelle elle code. Durant le second pas de temps, le processus de compétition et de persistance filtre les données de la première

étape. Ce n'est qu'après ces deux pas de temps que l'unité peut proposer son état à ses unités efférentes. La partie inférieure droite montre l'activité des unités U_A, U_B et U_C au cours du temps. La règle d'apprentissage hebbienne assure que si 2 unités sont actives en même temps, alors le poids qui les lie va croître. C'est le cas entre U_A et U_C à t et entre U_B et U_C à $t+n$. De cette façon, si la sélectivité de l'unité U_C est faible, alors U_C s'activera de façon causale à l'activation de U_A ou à l'activation de U_B. U_C réalise donc une association entre U_A et U_B.

Cette propriété permet d'obtenir des associations temporelles et des associations croisées multi-modales. Des unités sélectives à des propriétés de l'espace indépendamment du point de vue de l'observateur vont correspondre à des associations temporelles. Des unités dont l'activité est relative à une propriété de l'environnement indépendamment du point de vue et de la modalité d'observation vont correspondre à des associations croisées. Par exemple, le fait que des données brutes relatives à l'image ou au son produit par un objet activent la même unité. Cette propriété procure à l'agent la capacité d'abstraire des concepts à partir de données brutes particulières (Voir Figure III.9).

FIG. III.9 – Extraction de concepts, des perceptions particulières conduisent à l'activation d'unités communes.

III.5.6 Superposition des états

Nous avons également vu dans l'analyse qu'une même unité de traitement pouvait être amenée à participer au codage de plusieurs propriétés distinctes. Pour cette raison, il est suffisant de considérer que chaque propriété soit représentée par la synchronie des unités de traitement qui permettent de coder cette propriété.

Pour modéliser cette propriété nous découpons le pas de temps en plusieurs parties (Voir Figure III.10). Il est alors nécessaire que l'agent possède un système attentionnel qui permet d'acquérir séquentiellement des informations sur des parties très précises de l'environnement. Ce mécanisme attentionnel devra uniquement utiliser des informations de saillance (forme, couleur, mouvement, son).

Le modèle que nous avons présenté n'est pas totalement adapté à la propriété de superposition, en particulier la façon dont la propriété de persistance est modélisée. Pour être adapée, il faudrait avoir un compteur par sous-pas de temps.

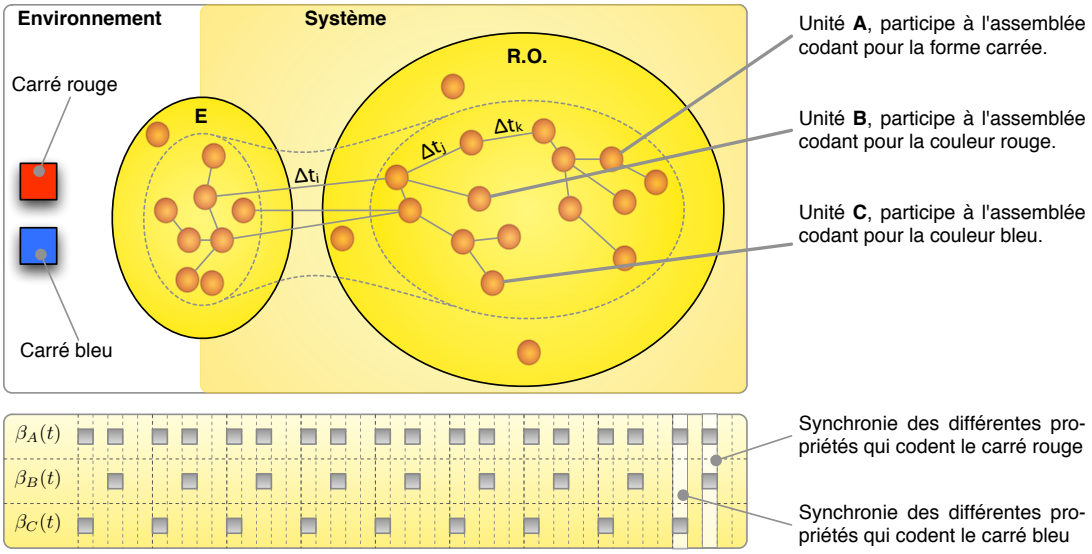


FIG. III.10 – Modélisation de la propriété de superposition des états.

III.6 Flux d'information subjective

L'état subjectif est responsable de l'apprentissage de la fonctionnalité. L'agent est hédoniste et cherche en agissant à activer les unités de traitement dont la composante subjective va permettre de faire croître l'état subjectif global. Chaque unité de traitement perçoit et participe à l'état subjectif global (Voir Figure III.11). L'état subjectif global est la moyenne de toutes les composantes subjectives des unités qui ont été activées ($\beta_i(t) > 0$). L'état subjectif global est également sujet à une fuite notée α_E. Il s'agit d'une modélisation linéaire de la propriété hédoniste globale. Encore une fois c'est un choix, et il y a certainement beaucoup de façons de modéliser cette propriété.

Nous appellerons ε_g l'état subjectif global et ε_i^l la composante locale subjective de l'unité U_i. Nous rappelons également que l'état objectif à un instant donné $EO(t)$ correspond à l'ensemble des unités actives à cet instant. Pour N unités de traitement dans le système de représentations objectives, nous avons alors :

$$\varepsilon_g(t+1) = L^-_{\alpha_E}\left(\varepsilon_g(t), \frac{1}{N}\sum_{U_i \in EO(t)} \varepsilon_i^l(t)\right) \tag{III.22}$$

Et l'évolution individuelle des $\varepsilon_l^i(t)$ est donnée par :

$$\varepsilon_i^l(t+1) = L^-_{\alpha_\varepsilon}\left(\varepsilon_i^l(t), \varepsilon_g(t) - \varepsilon_g(t-1)\right) \tag{III.23}$$

Les évolutions de l'état subjectif local et global sont liés. Quand une unité est activée, sa composante subjective tend vers la variation de l'état subjectif global avec le paramètre α_ε.

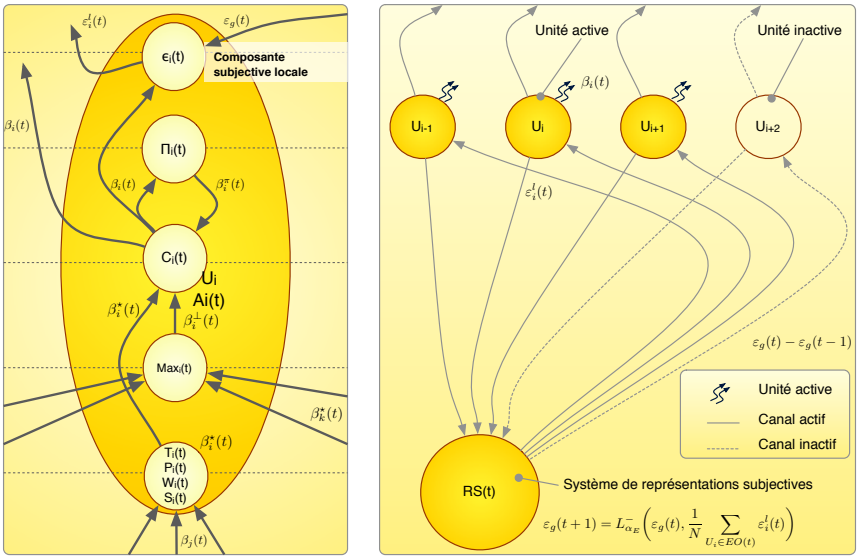


FIG. III.11 – Unité de traitement complète.

III.7 Organisation en réseau : Propriétés de groupe

Intéressons-nous maintenant aux propriétés de groupe des unités de traitement. Les unités de traitement sont organisées en réseau. Les propriétés de catégorisation reposent sur les réseaux d'unités de traitement. Les entrées du système sont des images et respectent ainsi la propriété de codage par position. D'après l'analyse, le système doit traiter les lieux et les formes de façon distribuée pour pouvoir généraliser ses comportements de façon efficace. Les données brutes sont des images. Pour être adaptées à ces images, les unités de conversion sont organisées en cartes bi-dimensionnelles. Par suite les premières couches qui observent les unités de conversions sont aussi organisées en cartes. De plus une unité de traitement qui observe des unités de conversion (champ récepteur) doit observer une région connexe dans ces données. Deux unités voisines observeront des régions voisines. Nous avons également montré dans l'analyse qu'un tel système doit agir pour percevoir et être en mesure de déplacer ses capteurs. Une même propriété locale peut donc se retrouver n'importe où dans les images constituées par les données brutes.

Pour un souci d'efficacité calculatoire, nous conférons aux premières couches la propriété d'invariance à la translation en leur permettant de partager leurs vecteurs de poids W. Nous avons vu que chaque région des données brutes était observée par un groupe d'unités de traitement. Nous choisissons de regrouper les unités de traitement en cartes (Voir Figure III.12). Chaque carte observe la même propriété dans l'image de données brutes *i.e.* possède le même vecteur de poids. De fait plusieurs cartes observent l'image de données brutes et sont en compétition pour ne pas extraire la même donnée (Voir Figure III.13). Si une unité gagne

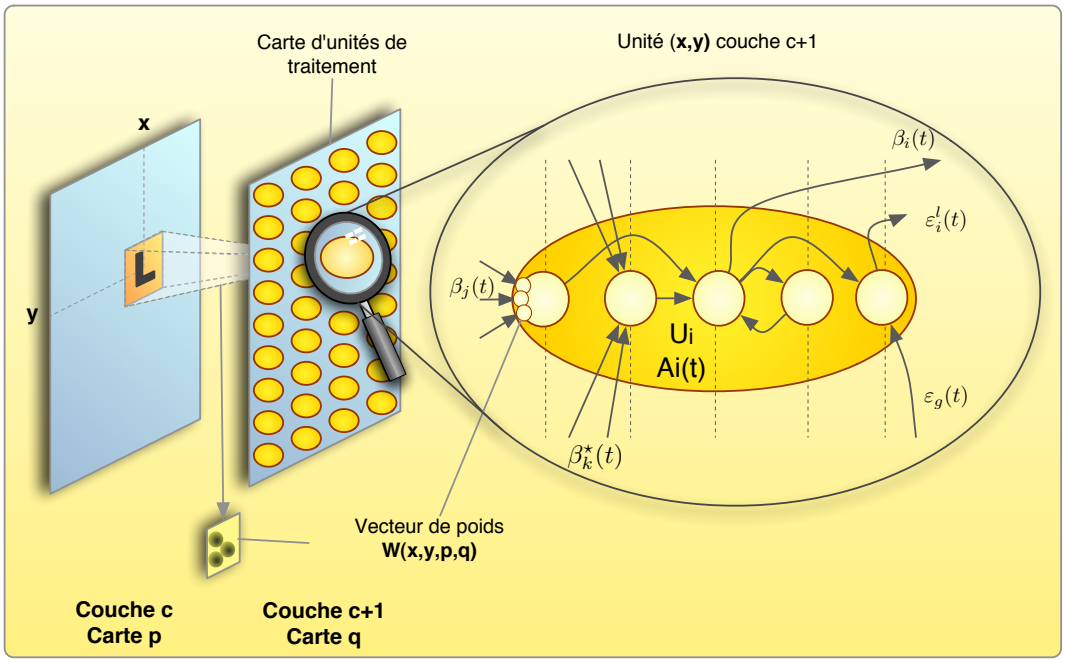


FIG. III.12 – Organisation en cartes.

une compétition, elle empêche alors les unités voisines de sa position dans les autres cartes de s'activer, de fait, elle les empêche aussi d'apprendre un motif. Au fur et à mesure de l'évolution du système les unités vont de plus en plus se spécialiser dans l'extraction de catégories particulières et différentes entre elles.

La Figure III.13 gauche illustre un cas simple où la première couche extrait des orientations locales. Chaque motif (**L**) qui est présent sur l'image de données brutes est ainsi décomposé dans le prisme perceptif. Cette décomposition est divergente entre la première couche et la seconde, elle va d'un ensemble de données indépendantes vers plusieurs catégories caractérisant des motifs dans les données brutes. La seconde couche observe la première couche et réitère le même traitement, c'est à dire qu'elle cherche des motifs stables pour les proposer à une éventuelle troisième couche. Ce processus d'apprentissage est à la fois spatial et temporel. Si les motifs sont présentés séquentiellement, chaque carte va se spécialiser dans l'extraction d'un motif particulier.

La partie droite de la Figure III.13 illustre la propriété de compétition spatiale. Dans ce cas deux motifs sont présentés (**L** et **T**). Il faut donc au moins 2 cartes pour séparer ces motifs en deux lieux différents. Lorsqu'une unité de la seconde couche gagne une compétition, elle empêche les autres cartes en compétition avec elle d'apprendre les motifs d'entrées au voisinage de cette position.

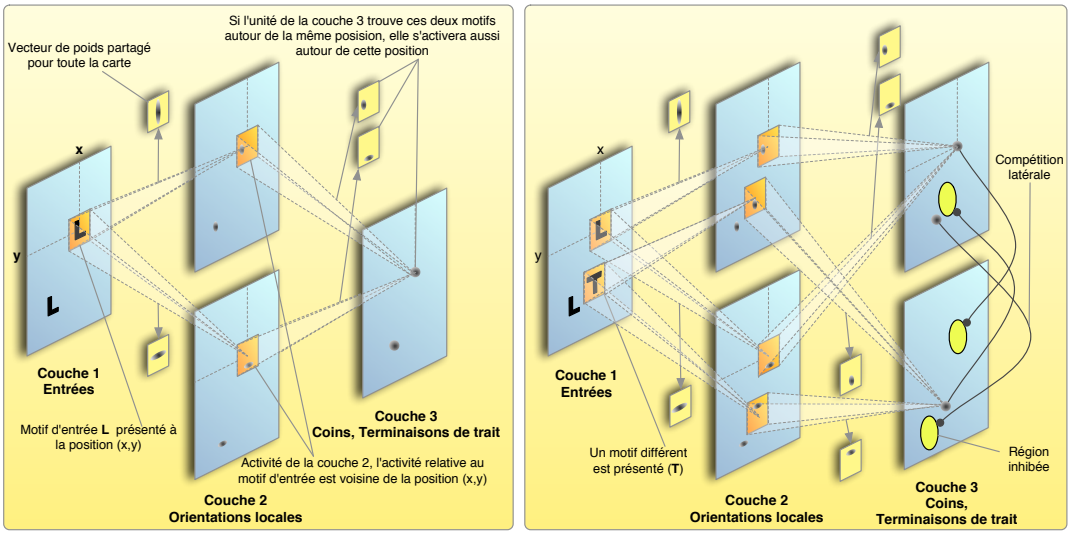


FIG. III.13 – Compétition spatiale.

III.8 Conclusion

Nous avons construit l'unité de base en lui donnant les propriétés nécessaires ou suffisantes que l'analyse à permis d'exhiber. Le modèle que nous proposons dans ce chapitre n'est pas unique. Il est basé sur un modèle de temps discret et permet une implémentation « event-driven » très efficace. Le fait de considérer le temps comme discret introduit un bruit dans le modèle. Il serait cependant possible d'avoir un meilleur modèle en considérant un modèle de temps continu. Dans ce cas la compétition pourrait être effectuée en utilisant l'ordre relatif des états d'activation. Un tel modèle est adapté à une machine de simulation unique, mais lorsque le calcul est réparti sur un réseau de machines, il devient difficile de faire en sorte qu'elles aient un temps commun. Nous présentons dans le chapitre suivant un simulateur de ce modèle.

Chapitre IV

NeuSter for dummies : Implémentation

NeuSter est un simulateur d'unités de traitement telles qu'elles ont été définies au chapitre précédent. Il est pour autant possible de tester n'importe quel modèle qui correspondrait au paradigme de codage par position. NeuSter est un ensemble de logiciels fonctionnant en parallèle sur des clusters hétérogènes de machines POSIX[1]. Comme nous l'avons vu, un système de codage par position est composé d'un grand nombre d'unités élémentaires, et d'un nombre fini d'algorithmes. Chaque unité peut prendre plusieurs valeurs, dont une particulière qui doit signifier le vide. Cette propriété est le « zéro » du système de codage par position. La notion de vide porte de l'information : le fait qu'il n'y ait pas la propriété représentée par cette unité.

IV.1 Architecture logicielle

IV.1.1 Présentation générale

Le modèle que nous présentons s'adapte parfaitement à un traitement parallèle. NeuSter est la contraction de « neuron » et de « cluster ». Il s'agit d'un simulateur de réseaux d'unités de traitements parallèle et interactif. La plupart des systèmes de calcul parallèle consistent à soumettre un job à un cluster de machine. Dans ce cas, l'utilisateur perd la main sur le système. NeuSter fonctionne en mode interactif, pendant une simulation, chaque paramètre peut-être modifié en temps réel. Pour avoir une idée de la taille de ce logiciel, il est composé de plus de 60 000 lignes de code C++, ObjectiveC, Perl et TCL, ce qui représente 1600 pages au format de cette thèse. Notre objectif n'est pas de proposer un manuel d'utilisation de NeuSter, mais

[1] Aujourd'hui, il a été compilé et testé sur MacOSX, Solaris, Linux, FreeBSD, et IRIX

de décrire brièvement la suite de logiciels qui le constituent et l'intérêt d'un tel simulateur, en particulier les choix et astuces qui permettent de traiter aujourd'hui sur des ordinateurs bon marché des réseaux de plusieurs millions de neurones et milliards de connexions en temps réel. NeuSter est très souple. Il est possible de modifier en ligne tous les paramètres d'un réseau. Créer des cartes, modifier ses différents paramètres, connecter ou déconnecter des cartes dynamiquement. Charger un réseau, sauver un réseau. Il y a plus de 120 commandes possibles. NeuSter est également scriptable en Perl et en TCL. Il est actuellement composé de 9 logiciels distincts (Voir Figure IV.1). Décrivons succinctement le rôle de chacune de ces composantes.

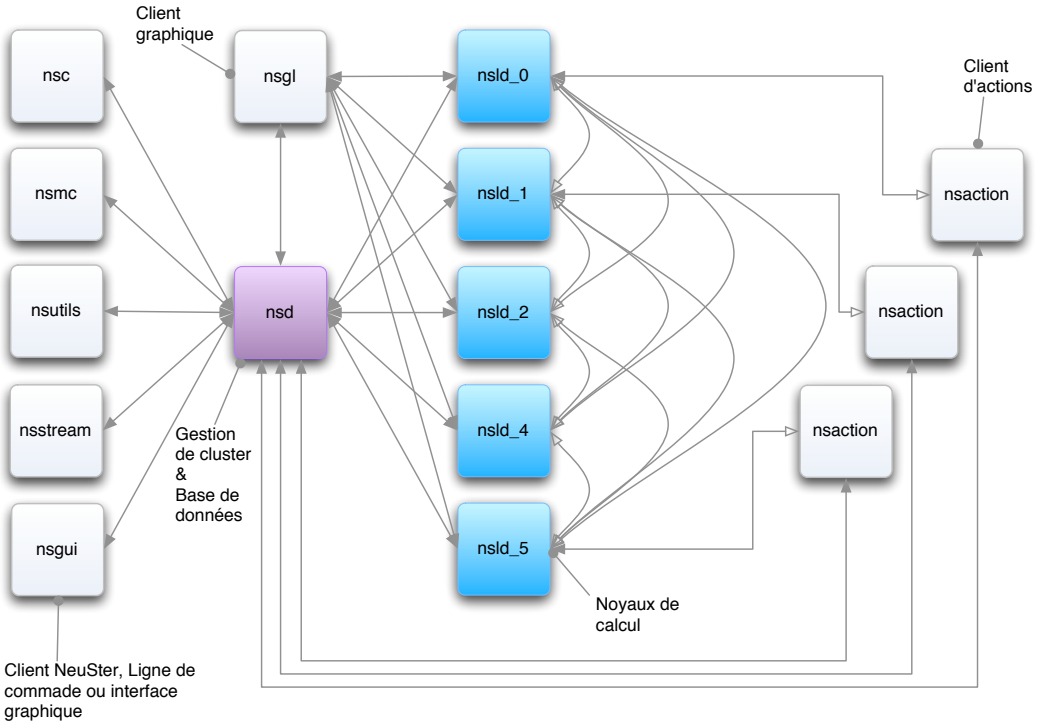


FIG. IV.1 – Ensemble de la suite de logiciels qui constituent NeuSter.

nsd (pour neuster server daemon) qui est le centre du système. Ce logiciel maintient une base de données sur les différents noeuds de calcul et les réseaux de cartes d'unités de traitement qui y sont déclarés. Il s'occupe également d'ordonnancer les pas de temps de calculs.

nsc et **nsmc** (pour neuster client et neuster multi-client) sont des clients nsd en mode ligne de commande. Les réseaux peuvent être créés soit avec des scripts Perl ou TCL, soit à l'aide de nsc.

nsld (pour neuster slave daemon) est le logiciel qui contient le noyau de calcul. C'est dans ce logiciel que les cartes d'unités de traitement sont créées et que leur état est calculé à chaque pas de temps.

nsstream est un logiciel qui permet d'envoyer un flux de données aux noyaux de calculs sous forme d'images (capture d'images, sonagramme, etc.)

nsaction est une interface entre un effecteur et une carte d'unité de traitement. Ce logiciel s'abonne à l'activité générée par une carte et agit en fonction des unités actives.

nsgl est un client d'affichage de l'état des objets. Si l'utilisateur le souhaite, il peut demander à un objet, c'est à dire une carte ou un vecteur de poids de s'afficher dans nsgl. Les images sont générées dans les noyaux de calcul et peuvent être acheminées directement à nsgl ou être routées par nsd. Comme nous l'avons vu, les unités de traitement sont composées de plusieurs variables d'état, entrées et sorties. Ces images sont construites à partir de l'ensemble des paramètres d'une carte donnée (potentiel, seuil, sorties, poids synaptiques, variance des données d'entrée, etc.). nsd n'a pas d'interface utilisateur, c'est un serveur qui attend des requêtes sur un port particulier. Le fait de découpler totalement nsd et les différents nsld d'une interface graphique permet de faire tourner ces logiciels sur des machines sans écran, ce qui est souvent le cas des clusters de calcul. L'interface de neuster peut parfaitement tourner sur une autre machine.

nsgui est un logiciel qui sert d'interface utilisateur à nsd. Il est écrit en TCL/Tk et est facilement modifiable. nsgui se connecte à un noyau nsd et propose à l'utilisateur de modifier en ligne chaque paramètre du réseau et contrôle la simulation (avance, pause, pas à pas). Il est également possible de sauver toutes ces images sur disque. Toutes les commandes qu'adresse nsgui à nsd sont également adressables en ligne de commande ou par script. Enfin

nsutils est un outil qui permet d'administrer le cluster, de connaître la performance de chaque machine, de savoir combien d'unités de traitement et de connexions sont mises à jour chaque seconde sur les différentes machines du cluster. L'ensemble de ces logiciels et la façon dont ils communiquent sont présentés Figure IV.1. Le processus de création de réseau consiste à démarrer nsd et charger une liste de noms de machines susceptibles de faire partie du cluster. Une commande particulière permet de créer de nouveaux noyaux de calcul (nsld). Ce processus est réalisé à l'aide d'outils de contrôle distant standard tel que *ssh*. Un simple script de quelques dizaines de lignes permet de déployer un réseau de plusieurs millions d'unités de traitement sur plusieurs machines et en quelques secondes. Nous détaillerons un exemple de script en fin de chapitre.

L'apprentissage d'un réseau peut être relativement long, en particulier pour les couches les plus hautes du réseau. L'apprentissage par expérience demande beaucoup d'essais pour que des composantes sensori-motrices indépendantes émergent. Pour cette raison l'état d'un réseau peut être sauvegardé sur disque et être rechargé plus tard.

Le fait que NeuSter soit composé d'un grand nombre de logiciels indépendants permet une plus grande robustesse de fonctionnement. Si une des composantes est inopinément interrompue, cela n'affecte pas les autres processus et ne stoppe pas la simulation. Par exemple, les composantes d'interface (nsgui, nsgl) sont complètement dissociées des composantes de calcul (nsd, nsld) et ne peuvent pas interrompre une simulation si elles viennent à être interrompues.

IV.1.2 Communication : système de messages

NeuSter fait un usage intensif du réseau[2]. Chaque composante logicielle peut fonctionner sur une machine différente (x86, PowerPC, Sparc) et chaque machine peut fonctionner sous un OS différent pourvu qu'il soit POSIX (Linux, MacOS X, Solaris, FreeBSD, IRIX). Il est donc facile de répartir la charge de calcul sur un cluster hétérogène et d'avoir une interface sur une simple machine de bureau.

Il y a deux types de flux d'information. Le premier flux est relatif à l'administration (création d'objets, création de connections, modification des paramètres relatifs aux objets crées), au contrôle des simulations (nouveaux pas de temps, fin de pas de temps), aux états subjectifs ($\varepsilon_g(t)$ et $\varepsilon_i^l(t)$ dans le modèle) et aux images des états des cartes si nécessaire. Le second flux est relatif aux données que s'échangent les cartes (L'état des sorties $\beta_i(t)$ des différentes unités de traitement du modèle que nous avons présenté).

Pour cela nous avons développé un système de messages indépendants du type de processeur et du système d'exploitation sur lequel tourne chacune des composantes logicielles. Ce système de message est basé sur la sérialisation de classe. Nous avons défini un ensemble de messages élémentaires qui, interprété sur une machine cible, permet d'exécuter une fonction distante associée à un ensemble d'arguments. Cette façon de procéder est très proche du système RPC[3]. Ce système de message a été développé pour qu'aucune copie ne soit nécessaire. Les données qui arrivent sur une machine distante sont directement exploitables.

Les messages sont tous construits sur le même modèle : ils disposent d'une taille, du pas de temps où ils ont été émis, de l'identifiant de l'objet d'origine, et de l'action à réaliser éventuellement associée à un ensemble de données. Les actions élémentaires sont choisies de telle façon que les données qui circulent soient les plus simples possible, c'est-à-dire une valeur ou un tableau de valeurs.

Les messages sont sérialisés et partent d'une boîte de message pour aller vers une autre. Les boîtes de messages envoient les messages par paquets en utilisant le protocole TCP/IP. Ce protocole garantit qu'aucun message ne peut se perdre durant la simulation. La communication entre deux composantes tournant sur la même machine est également réalisée de cette façon. Chaque composante possède plusieurs boîtes de messages. nsd possède un canal de communication pour chaque noyau de calculs (nsld) participant à la simulation, un canal dédié aux commandes et requêtes utilisateur et un canal optionnel vers (nsgl) qui permet de router les paquets d'images de l'état du réseau. les noyaux de calcul sont connectés à nsd et établissent des connexions entre eux si nécessaire pour échanger leurs changements d'état.

La cause principale de l'efficacité de NeuSter réside dans l'aspect « event-driven » de la propagation des traitements. Ceci est rendu possible par l'existence d'un état d'activation neutre de l'unité de traitement. Nous allons maintenant détailler ce procédé.

[2]Nous utilisons l'API socket BSD, en particulier le protocole TCP/IP. Cela permet de distribuer une simulation sur un réseau local de machines, ou même à travers internet

[3]Remote Procedure Call

IV.2 Algorithmique : L'aspect « event driven »

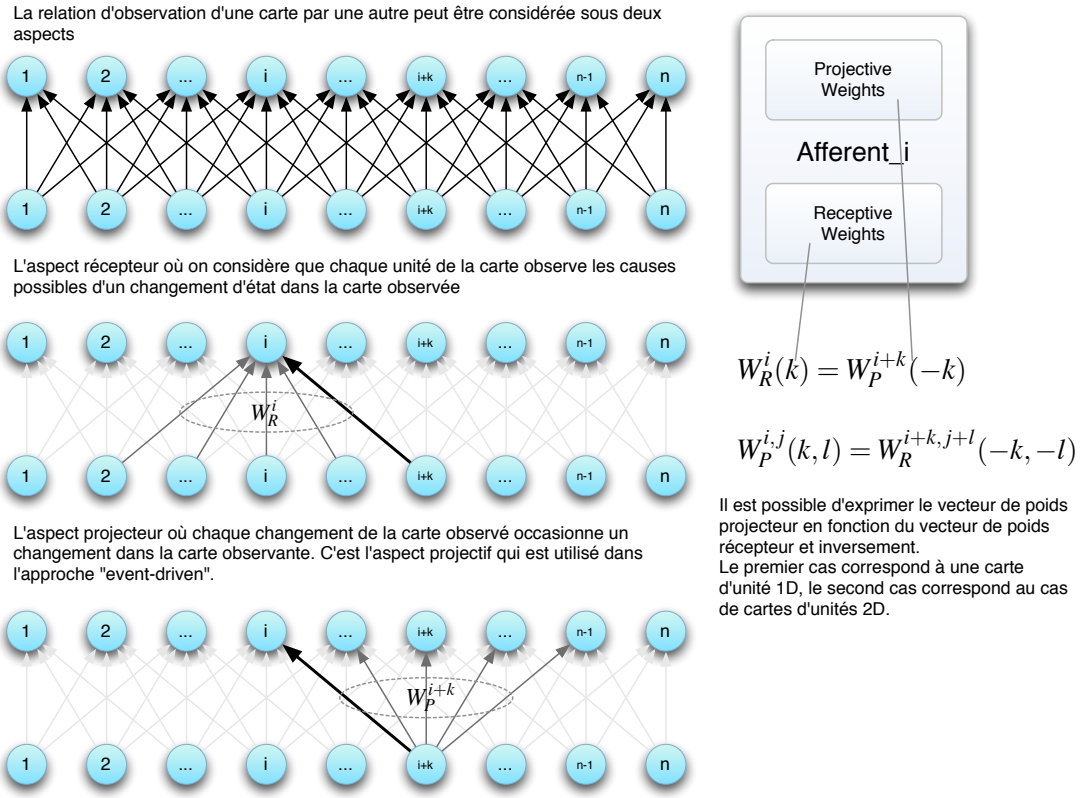


FIG. IV.2 – Passage des vecteurs de poids réceptifs aux vecteurs de poids projectifs.

Il y a deux façons de mettre à jour l'état des unités de traitement. La plus classique consiste à faire en sorte que chaque unité observe à chaque pas de temps l'état de ses afférents. Le nombre d'observation qui va être réalisé à chaque pas de temps correspond à la somme du nombre d'afférent de chaque unité de traitement. Soit N le nombre total d'unités de traitement et $Card(\Omega_i)$ le nombre d'afférent de l'unité U_i. Alors, traiter la mise à jour de l'état des unités de traitement va demander $\sum_{i=0}^{N} Card(\Omega_i)$.

L'autre façon de réaliser la mise à jour de l'état d'activation des unités de traitement consiste à propager l'état des unités à l'ensemble de ses efférents. Cette approche est dite « event-driven ». À chaque pas de temps, on met à jour l'état des entrées du réseau et on propage cet état dans tout le réseau. Si on appelle Λ_i l'ensemble des connexions efférentes de l'unité U_i, alors en traitant le problème de cette façon, il y aura $\sum_{i=0}^{N} Card(\Lambda_i)$ unités à mettre à jour.

Si on ne s'intéresse pas à l'état des unités, alors ces deux quantités de calculs sont exac-

tement les mêmes. Le fait de disposer d'un état d'activation nul dans notre modèle donne un avantage très important à la seconde approche. En effet, si une unité n'a pas été touchée par les informations qu'elle attendait, alors elle restera inactive (sauf si elle est dans un état de persistance). De ce fait, pour chaque unité inactive, un calcul sera économisé pour chaque connexion formée par le cône de ses efférents. Si on considère qu'en moyenne seule 2% à 5% des unités sont actives alors les mêmes calculs pourront être réalisés 20 à 50 fois plus rapidement. De ce fait, s'il n'y a aucune unité active, alors il n'y a pas de calcul.

Cette façon de propager les changements d'état dans le réseau demande de présenter les vecteurs de poids de façon adaptée. Le modèle que nous avons présenté au chapitre précédant considère que chaque unité possède un vecteur de poids récepteur W. Il est possible de transformer ces vecteurs de poids réceptifs en vecteurs de poids projectifs. C'est ce qu'illustre la Figure IV.2 dans le cas de cartes unidimensionnelles. Cette transformation se généralise facilement au cas de cartes bidirectionnelles. L'ensemble des paramètres qui caractérisent une connexion entre deux cartes est localisé dans un objet appelé $afferent$. Les vecteurs de poids réceptifs sont utilisés pour l'apprentissage des causes de l'activation et les vecteurs de poids projectifs sont utilisés pour l'intégration des informations.

IV.2.1 Cartes d'unités et listes d'événements

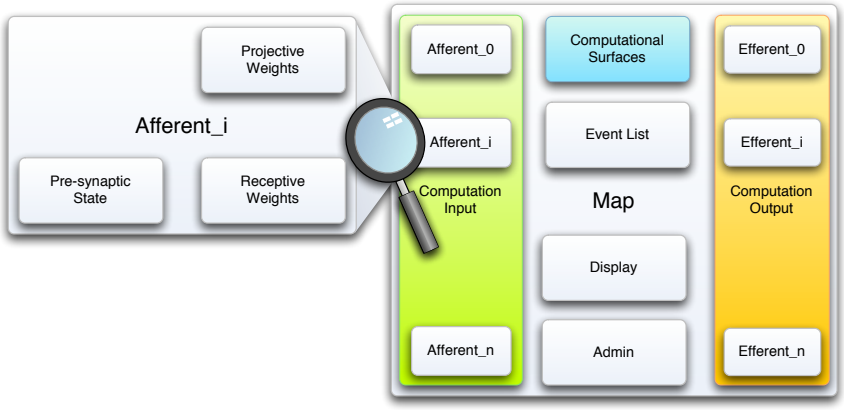


FIG. IV.3 – L'objet Map.

Un réseau sera décrit comme un ensemble de cartes d'unités de traitement réparties sur différents noeuds de calculs (nsld) et la façon dont ces cartes seront connectées. Chaque carte possède un ensemble d'afférents et d'efférents, l'ensemble des paramètres qui caractérisent chaque unité, et un ensemble de systèmes de communications (Voir Figure IV.3). Il y a 3 flux de communication : un flux dédié aux messages d'administration et de modifications des différents paramètres de la carte, un flux de calcul qui lie les différents noyaux et permet aux cartes d'in-

former leurs efférentes que leur état a changé, enfin il y a un flux dédié à l'affichage[4]. Chaque objet *a f f erent* relatif à une carte comporte un vecteur de poids réceptif et projectif ainsi qu'une carte représentant l'état d'activation de la carte observée (c'est ce que nous appelons *S* dans le modèle). Lorsqu'une unité apprend la cause de son activation, elle modifie ses poids réceptifs en fonction de l'état des unités observées. Les différentes cartes constituant un réseau sont susceptibles de fonctionner sur différentes machines. Il est donc préférable de conserver une image d'état pré-synaptique (*S*) localement si on souhaite minimiser la quantité de messages réseau nécessaire pour retrouver l'état des afférents.

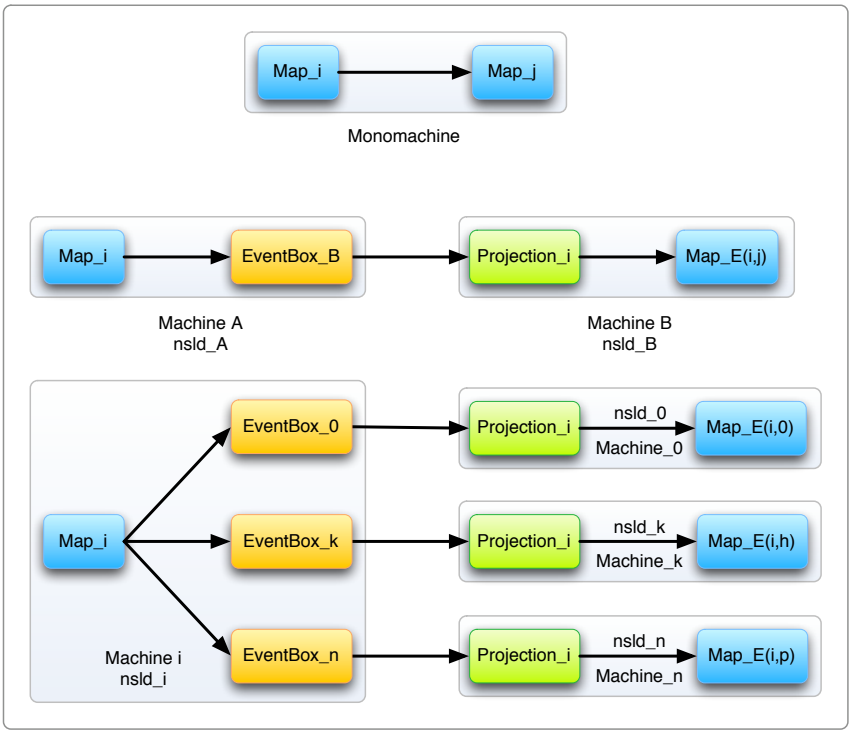


FIG. IV.4 – Algorithme distribué.

Les paramètres caractérisant l'état des unités de traitement sont rassemblés en sous-cartes homogènes. Une carte de potentiel, une carte de seuil, une carte d'état d'activation, une carte de persistance, une carte de composante subjective. Cette organisation est beaucoup plus adaptée aux calculs que nous devons réaliser que celle qui consiste à considérer un objet relatif aux unités de traitement.

Ce qui caractérise l'état d'une carte à chaque pas de temps c'est l'ensemble des unités de traitement actives. La propagation de cette information peut donc se résumer à des listes de co-

[4]Le flux d'affichage est optionnel, et n'a pas d'utilité computationnelle.

ordonnées d'événement dans le réseau. Chaque carte possède un ensemble de cartes efférentes. Elle va donc générer et envoyer la liste des événements qui caractérise son état à ces cartes efférentes. Ces listes d'événements sont des messages particuliers qui ne circulent qu'entre les noyaux de calcul (nsld). Une liste de message est toujours constituée d'une date d'émission, d'une origine, d'une taille et de la liste des coordonnées des modifications d'état d'activation de la carte d'origine. Chaque coordonnée est associée à une valeur de sortie $(i, j, \beta_{ij}(t))$.

Lorsqu'une carte envoie une liste d'événements à l'une de ses cartes efférentes, elle commence par sérialiser une liste d'événement. Cette liste est déposée dans la boite d'envoi relative aux noyaux où se trouve la carte destinataire (Voir Figure IV.4) Si la carte destinataire se trouve sur le même noyau, nous utilisons la même méthode, simplement le message passera par l'interface réseau appelée *localhost* et pas par le réseau. Les listes d'événement relatifs à un noyau distant sont envoyés en une seule fois. Nous devons attendre que toutes les cartes d'un noyau donné qui génère une liste pour un noyau distant aient terminé leur calcul pour envoyer ces messages liste. Lorsque les messages arrivent sur un noyau distant, un représentant de la carte d'origine appelé $projection$ filtre les listes de messages. Il utilise le champ origine du message pour déterminer si c'est à lui de réaliser l'intégration. Les cartes qui vont être intégrées à l'aide de cette liste sont toutes les cartes efférentes de la carte d'origine se trouvant sur ce noyau. Elle vont toutes intégrer la même liste (elle observent toutes les mêmes données provenant de la carte d'origine), mais elles utiliseront chacune leur propre vecteur de poids pour en extraire des informations particulières.

IV.2.2 Algorithme parallèle de calcul

L'algorithme général de fonctionnement de NeuSter est découpé en pas de temps. Il est réparti sur plusieurs machines et chaque noyau de calcul est $multi-thread$. Lorsque le réseau est créé, nsd envoie un message de démarrage à tous les noyaux de calculs. Chaque noyau de calcul est composé de deux *thread* . Le premier écoute nsd et attend l'ordre de démarrage : nous appelons ce *thread* **Fire**. Il correspond au tir des listes d'événements lorsque les unités se sont activées. Le second *thread* est appelé **Integrate**. Ce *thread* écoute les messages en provenance des autres noyaux de calcul et traite exclusivement des listes d'événements. La partie gauche de la figure IV.5 illustre les échanges de messages entre nsd et les différents *threads* de nsld.

Chaque carte connaît son nombre d'afférents par construction. Lorsqu'un noyau a intégré autant de listes d'événements que la somme des afférents des cartes qu'il contient, il répond à nsd pour lui confirmer qu'il a réalisé tous ses calculs pour le pas de temps en cours. De la même façon nsd connaît le nombre total de noyaux relatifs à une simulation. Lorsque tous les noyaux ont terminé leur calcul, nsd relance un nouveau pas de temps de calcul et ainsi de suite.

Il faut faire attention à un point particulier dans l'ordonnancement des *threads* **Fire** et **Integrate** pour chaque noyau de calcul. Au cours d'un pas de temps, une intégration ne doit pas commencer si la carte n'a pas tiré sa liste d'événement. Si le *thread* de tir avait lieu après l'intégration, l'état des unités de traitement serait différent. La partie droite de la figure IV.5 illustre un cas où le *thread* d'intégration doit attendre la fin du *thread* de tir avant de débuter.

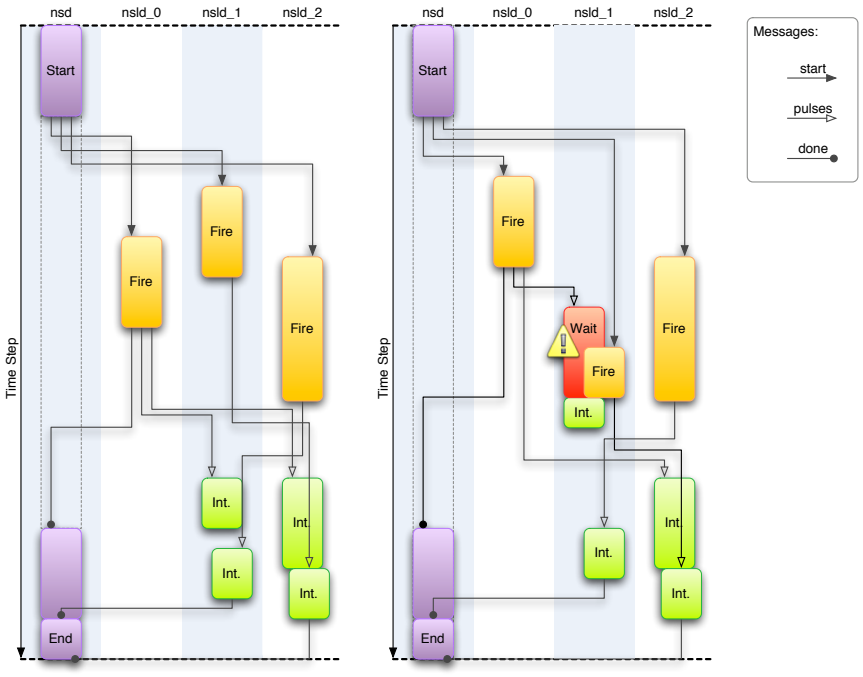


FIG. IV.5 – Algorithme distribué.

IV.2.3 Intégration des listes d'événements

Lorsqu'une liste d'événements est intégrée par une carte, le vecteur de poids projectif est utilisé. L'algorithme d'intégration revient à ajouter pour chaque coordonnée dans la liste d'événements le vecteur de poids de l'unité touchée autour de cette coordonnée. Ce procédé est illustré par la Figure IV.6. On comprend à l'aide de cette figure que l'intégration d'une liste concerne la coordonnée touchée ainsi que son voisinage. C'est la raison pour laquelle l'unité computationnelle élémentaire est la carte. Distribuer une carte entre plusieurs noyaux produirait de nombreux effets de bords lors de l'intégration et augmenterait de façon significative la quantité de messages réseau nécessaires.

Dans le cas où les vecteurs de poids sont partagés pour l'ensemble des unités d'une carte, l'intégration revient à réaliser pour chaque pas de temps une convolution entre le vecteur de poids et l'ensemble des coordonnées de la liste d'événements et à ajouter ce résultat partiel à la carte de potentiel de la carte touchée.

Lorsque toutes les listes afférentes ont été reçues, la carte sera prête à tirer son état lorsque le prochain pas de temps débutera.

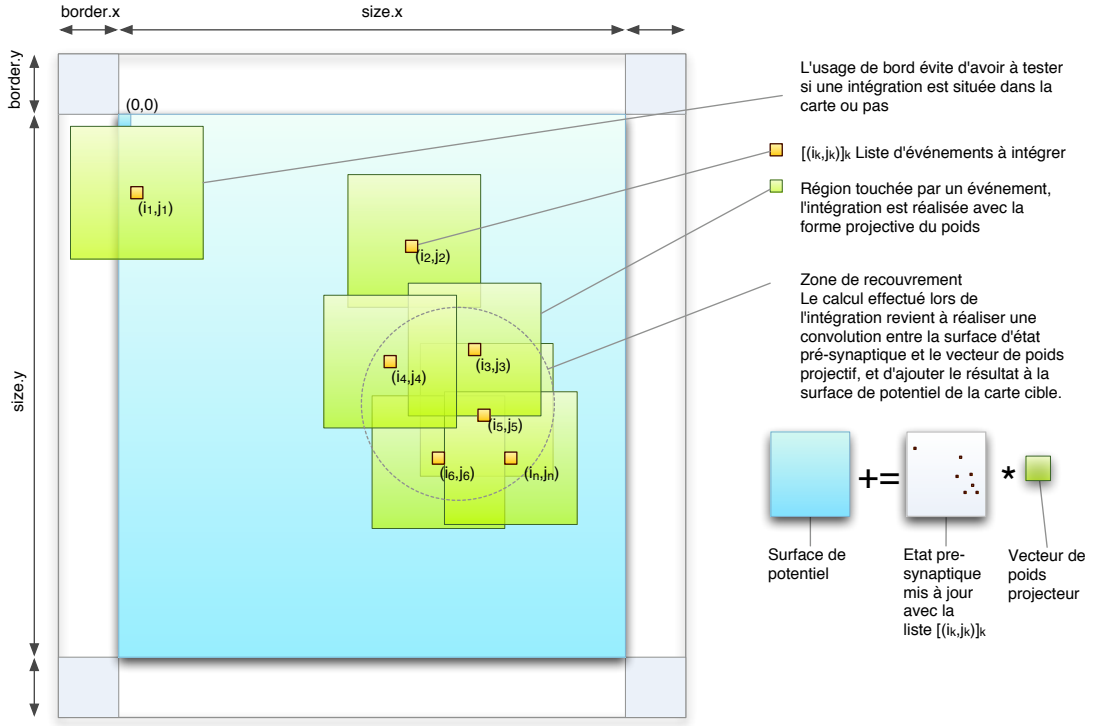


FIG. IV.6 – Intégration de listes d'événements.

IV.2.4 Algorithmes

L'algorithme général est relativement complexe. Nous allons le présenter en trois étapes. Dans un premier temps, nous présentons une version simplifiée qui ne comprend que le *thread* d'intégration et de tir. L'algorithme 1 présente la règle intègre et tire minimale. L'algorithme 2 présente la règle intègre et tire à laquelle est ajouté l'inhibition, enfin l'algorithme 3 est complet et comprend l'inhibition et l'apprentissage. Comme nous l'avons vu, cette approche est basée sur un principe de codage par position et est de fait caractérisée par un ensemble fini d'algorithmes.

IV.3 Entrées et sorties

NeuSter comprend un grand nombre d'entrées et de sorties potentiellement utilisables. Des bibliothèques d'entrées ont été développées pour permettre de réaliser des acquisitions à partir de caméras ou de microphones. Les acquisitions vidéo et audio dépendent des plates-formes utilisées, mais sont facilement extensibles. Sous GNU/Linux nous disposons de pilotes de caméra Firewire, sur MacOS X nous utilisons l'API QuickTime©qui permet d'utiliser

Data : L'état du réseau, les signaux d'entrée
Result : L'état du réseau, les signaux de sortie
Thread Fire();
foreach *Carte M dans le réseau* **do**
  foreach *Unité U de coordonnée* (i, j) **do**
    if $P_{ij} \geqslant T_{ij}$ **then**
      Calcul de la valeur de sortie $\beta_{ij} = F(P_{ij}, T_{ij})$);
      Ajout d'un événement (i, j, β_{ij}) à la liste $L(M,t)$;
    end
    Application de la fuite sur le potentiel P_{ij};
  end
end
Thread Integrate();
foreach *Liste d'événement L* **do**
  foreach *Carte M touchée par la liste L d'origine* $M^\star$ **do**
    foreach *Evénement de coordonnée (i,j) et de valeur* β_{ij} *dans L* **do**
      Extraction du noyau de poids projectifs $W_P^{i,j}(M, M^\star)$;
      Intégration du potentiel avec $W_P^{i,j}(M, M^\star)$;
    end
  end
end

Algorithme 1 – Algorithme minimal de type intègre et tire

Data : L'état du réseau, les signaux d'entrée
Result : L'état du réseau, les signaux de sortie
Thread Fire();
foreach *Carte M dans le réseau* **do**
  foreach *Unité U de coordonnée (i,j)* **do**
    if $P_{ij} \geqslant T_{ij}$ **then**
      if $P_{ij} >$ *niveau d'inhibition* I_{ij} **then**
        Calcul de la valeur de sortie $\beta_{ij} = F(P_{ij}, T_{ij})$;
        Ajout de cet événement (i, j, β_{ij}) à la liste $L(M,t)$;
      end
    end
    Application de la fuite sur le potentiel P_{ij};
  end
end
Thread Integrate();
foreach *Liste d'événement L* **do**
  foreach *Carte M touchée par la liste L d'origine* $M^\star$ **do**
    if *La connexion est* ***excitatrice*** **then**
      foreach *Evénement de coordonnée* (i, j) *et de valeur* β_{ij} *dans L* **do**
        Extraction du noyau de poids projectifs $W_P^{i,j}(M^\star, M)$;
        Intégration du potentiel avec $W_P^{i,j}(M^\star, M)$;
      end
    end
    if *La connexion est* ***inhibitrice*** **then**
      foreach *Evénement de coordonnée* (i, j) *et de valeur* β_{ij} *dans L* **do**
        Extraction du masque d'inhibition projectif $W_P^{ij}(M^\star, M)$;
        Intégration du niveau d'inhibition avec $W_P^{ij}(M^\star, M)$;
      end
    end
  end
end

Algorithme 2 – Algorithme intégration et inhibition.

Data : L'état du réseau, les signaux d'entrée
Result : L'état du réseau, les signaux de sortie
Thread Fire();
foreach *Carte M dans le réseau* **do**
  foreach *Unité U de coordonnée* (i, j) **do**
    if $P_{ij} \geqslant T_{ij}$ **then**
      if $P_{ij} > I_{ij}$ **then**
        Calcul de la valeur de sortie $\beta_{ij} = F(P_{ij}, T_{ij})$;
        Ajout de cet événement $(i, j, beta_{ij})$ à la liste $L(M,t)$;
        foreach *Pour chaque carte afférente* $M^\star$ *de M* **do**
          Extraction de l'état près-synaptique autour de (i, j);
          Mise à jour de noyau de poids réceptifs $W_R^{ij}(M^\star, M)$;
        end
      end
    end
    Application de la fuite sur le potentiel P_{ij};
  end
end
Thread Integrate();
foreach *Liste d'événement L* **do**
  foreach *Carte M touchée par la liste L d'origine* $M^\star$ **do**
    if *La connexion est* ***excitatrice*** **then**
      foreach *Evénement de coordonnée* (i, j) *et de valeur* β_{ij} *dans L* **do**
        Extraction du noyau de poids projectifs $W_P^{i,j}(M^\star, M)$;
        Intégration du potentiel de la carte touchée M avec $W_P^{i,j}(M^\star, M)$;
        Mise à jour de l'espace pré-synaptique à la coordonnée (i, j);
      end
    end
    if *La connexion est* ***inhibitrice*** **then**
      foreach *Evénement de coordonnée* (i, j) *et de valeur* β_{ij} *dans L* **do**
        Extraction du masque d'inhibition projectif $W_P^{ij}(M^\star, M)$;
        Intégration du niveau d'inhibition avec $W_P^{ij}(M^\star, M)$ de valeur β_{ij};
      end
    end
  end
end

Algorithme 3 – Algorithme intégration, inhibition et apprentissage.

n'importe quelle caméra USB ou Firewire, pour peu qu'un pilote soit présent sur le système utilisé. L'acquisition audio est également dépendante de la plate-forme. Nous travaillons avec /dev/audio sur GNU/Linux et Solaris. Sur MacOSX nous utilisons CoreAudio, qui est un ensemble de bibliothèques et d'extension noyau qui permet d'acquérir et de produire des sons.

Les bibliothèques d'actions permettent de piloter le module Pan/Tilt relatif d'une camera. Ceci permet de simuler des systèmes de vision active. Il est également possible de piloter un synthétiseur vocal sur GNU/Linux et MacOSX. L'architecture logicielle de NeuSter permet aisément d'étendre les bibliothèques d'entrées et de sorties du système. Chaque système d'entrée et de sortie doit obéir au principe de codage par position et si possible être représentable à plusieurs échelles spatiales.

IV.4 Création de réseaux

Les réseaux sont créés à l'aide de commandes particulières envoyées dans un ordre particulier à nsd. Ces messages sont le plus souvent des chaînes de caractère. Ces messages de création de réseaux peuvent être écrits dans n'importe quel langage compilé ou interprété. Nous avons choisi de construire des classes de création de réseau qui soient suffisamment simples et puissantes. Nous avons choisi Perl, TCL et Obective-C++, mais ce n'est qu'un choix qui pourrait aisément être étendu à d'autres langages. La figure IV.7 montre un exemple de réseau créé à l'aide d'un script Perl. Les étapes les plus importantes de l'exécution du script sont numérotées sur le schéma représentant l'ensemble des processus créés. La première commande crée nsd si il n'existe pas et charge la liste des machines appartenant au cluster que nous avons défini. Cette liste se trouve dans un fichier caché à la racine du répertoire personnel ($\tilde{/}$.neuster/machines.script). Les commandes suivantes créent respectivement le premier noyau de calcul (nsld_0), puis une source vidéo, une conversion avec réduction. La conversion est l'étape qui permet de passer d'un ensemble de mesures physiques aux signaux que s'échangent les unités de traitement ($\beta_i(t)$). Dans ce cas précis, les unités de conversion transforment la luminance d'un groupe de pixels en contrastes locaux positifs et négatifs. Les deux derniers blocs créent les autres noyaux de calcul, respectivement nsld_1 et nsld_2. Sur chacun de ces noyaux sont déclarés 4 cartes connectées aux cartes de conversion de nsld_0. Chaque carte utilise ici un schéma de vecteurs de poids partagé et observe les cartes de conversion de nsld_0 à l'aide de vecteurs de poids dont les valeurs sont issues de fonctions de Gabor. Les fonctions de Gabor permettent dans ce contexte d'extraire des orientations locales, en particulier des bords. Dans cet exemple particulier les 4 cartes de nsld_1 extraient des bords de 0° à 135° et les 4 cartes de nsld_2 extraient des bords de 180° à 315°.

Cet exemple illustre un réseau sans apprentissage. Si une carte apprend, nous initialisons simplement ses poids de façon aléatoire, le plus souvent en modulant un bruit blanc avec une gaussienne de façon à avoir des poids plus importants au centre. Nous détaillerons ces méthodes dans le chapitre suivant.

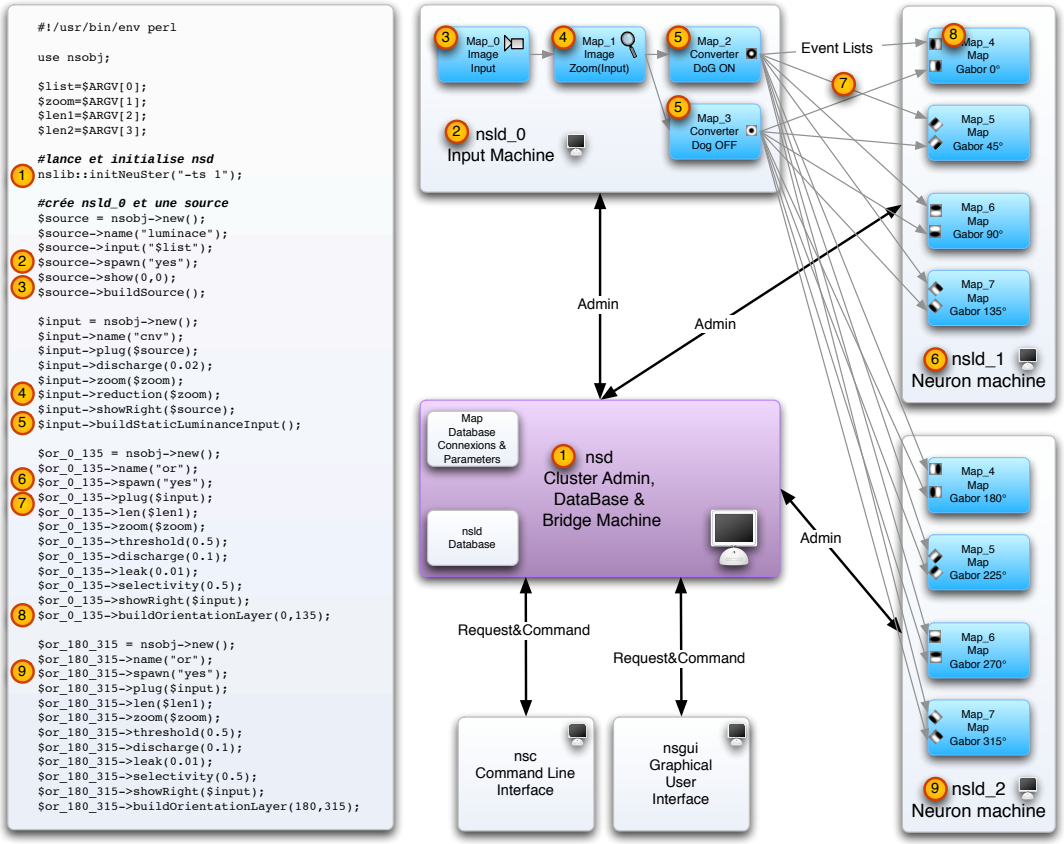


FIG. IV.7 – Script de création de réseau.

IV.5 Conclusion

Réaliser un système complet d'apprentissage ouvert de représentations et de fonctionnalités demande de simuler des millions d'unités de traitement. Ce type de système ne peut pas fonctionner sur une seule machine à l'heure actuelle, il est encore moins pensable de réaliser des simulations sous des logiciels de tests tels que Matlab©. Il est donc nécessaire de développer un logiciel spécifique pour simuler ce type d'agent. En ce sens NeuSter représente un premier résultat partiel. Il est maintenant possible de tester des hypothèses d'architecture très rapidement, NeuSter est facilement extensible. La partie qui consiste à administrer le cluster est clairement séparée de la partie simulation de modèle. Il est très simple de tester d'autres modèles. Le chapitre suivant présente un ensemble de réseaux simulés dont l'objectif est de montrer que le modèle développé au chapitre précédant possède bien les propriétés recherchées au chapitre II.

Chapitre V

Premiers résultats

Les premiers résultats que nous proposons dans ce chapitre visent à montrer que le modèle proposé au chapitre III possède bien les propriétés que nous avons exhibées au chapitre II, en particulier la capacité à former des catégories à travers une compétition temporelle ou spatiale et la capacité à traiter des données provenant de n'importe quelle modalité. Comme nous l'avons vu, ce modèle n'est pas unique. Les propriétés nécessaires ou suffisantes conduisent à une classe de modèles. Ces résultats visent à montrer le caractère général de l'approche.

Il est aujourd'hui relativement complexe de tester un système mettant à la fois en oeuvre l'apprentissage de représentations et l'apprentissage de fonctionnalités. La quantité d'opérations à effectuer chaque seconde est encore trop importante pour les machines dont nous disposons aujourd'hui à bord des robots. C'est pour cette raison que NeuSter a été développé. Ce simulateur permet de partager les calculs sur un cluster de machines et nous rapproche un peu plus du jour où un test global sera réalisable. L'apprentissage de fonctionnalités est bâti sur le système de représentation : nous allons donc essentiellement exposer des résultats relatifs à l'apprentissage de représentations.

Le principe de codage par position envisage la notion de sémantique sous un angle nouveau. L'infinité des sens est repoussée dans l'infinité des lieux. Le sens de chaque unité de traitement change en fonction de sa position dans le réseau. Quelle que soit la catégorie à représenter, il existera un réseau suffisamment grand pour la représenter : même les catégories dont on ignore préalablement l'existence peuvent être représentées par un tel système. En cela, les représentations exprimées forment un système de catégorisation ouvert. Nous entendons ouvert dans le sens où il sera possible pour le système d'appréhender la nouveauté et de l'intégrer à ses connaissances de façon autonome sans qu'il soit nécessaire de repenser le modèle de

fonctionnement.

Nous avons vu dans le chapitre III.2 que les entrées d'un système de codage par position devaient également respecter cette contrainte de codage par position. La question du sens ne se pose jamais explicitement. Les entrées du système sont composées d'unités de conversion qui réalisent la transformation de mesures physiques en informations exploitables par le réseau d'unités de traitement. Le choix de la représentation des données brutes joue ainsi un rôle très important.

V.1 Quels résultats attendre d'un système ouvert ?

Le système ouvert que nous étudions n'a aucun but préalablement fixé. Notre système de représentation ouvert doit apprendre à différencier des situations de l'environnement en construisant des catégories. Une situation complexe de l'environnement ne doit pas conduire à l'activation d'une catégorie unique, mais à l'activation d'une assemblée d'unités codant chacune pour une catégorie caractérisant une part de cette situation. Le fait que les représentations soient distribuées doit conduire à la capacité de généralisation du système. Pour une modalité sensorielle particulière, il n'y a aucune modélisation spécifique, le seul travail consiste à décrire un réseau d'unités de traitement.

Les réseaux les plus petits comptent plusieurs milliers d'unités de traitement. Chacune de ces unités est composée d'un ensemble de paramètres qui évoluent au cours du temps et codent une propriété particulière en fonction de sa position dans le réseau. La connectivité des unités de traitement est également très grande. L'état des unités doit être recalculé plusieurs fois par seconde pour que la boucle sensori-motrice du robot soit compatible avec une interaction humaine. Une unité observe plusieurs centaines d'autres unités et est à son tour observée par plusieurs centaines d'unités. On ne peut pas connaître à l'avance l'information que va extraire une unité particulière. Cette différenciation se fait au cours de mécanismes de compétition et à partir de valeurs initiales choisies aléatoirement. Lorsqu'une unité se sera spécialisée, alors son activation véhiculera toujours la même signification. Pour toutes ces raisons, il est difficile de présenter des résultats pour un tel système.

Ainsi, notre premier objectif n'est pas de démontrer la performance du modèle local, mais plutôt l'aspect ouvert de la méthode. Nous avons choisi une approche très classique pour la première étape de traitement des unités de traitement. Le traitement effectué par les unités consiste à séparer l'ensemble des configurations possibles de l'espace d'entrée à l'aide d'un hyperplan. Quelle que soit la catégorie à extraire, il existera un réseau suffisamment profond pour la représenter[1].

[1] Pour réaliser un **ou exclusif** (XOR) entre deux cas, il est nécessaire de disposer d'au moins 2 couches de traitements

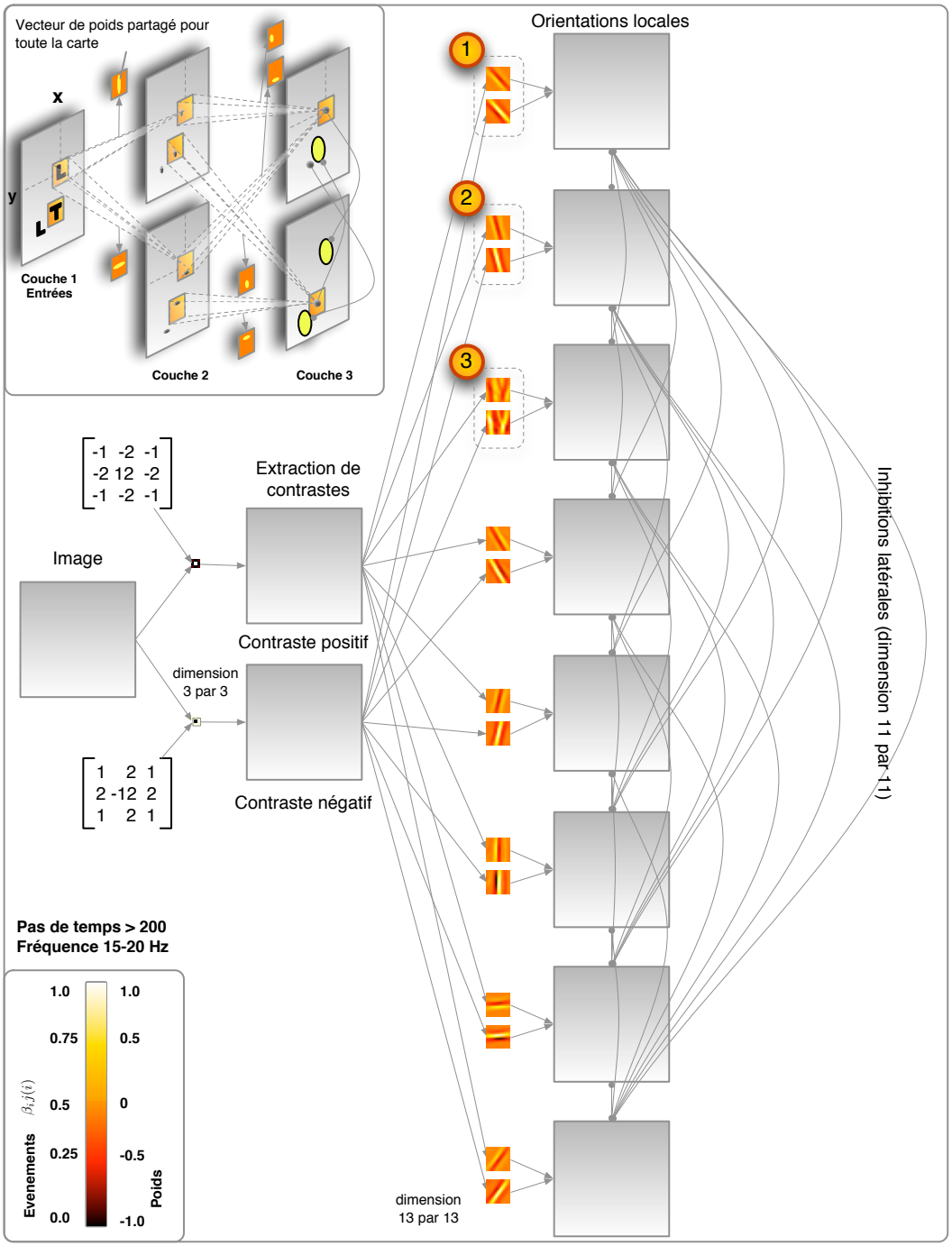


FIG. V.1 – Catégorisation et localisation de bords et traits.

V.2 Un système de représentation ouvert

Nous allons mettre en évidence l'aspect général de la méthode en montrant qu'il est possible de catégoriser n'importe quel type de données pourvu qu'elles respectent aussi la propriété de codage par position. Nous allons donc présenter un ensemble de réseaux simple réalisé à l'aide de NeuSter. Les scripts de création sont très proches de celui présenté à la figure IV.7 du chapitre IV. Il est possible d'extraire des orientations locales, des motifs de second ordre (des coins), des visages, des mouvements apparents, et enfin des phonèmes. L'extraction de ces propriétés ne nécessite aucune modélisation préalable et est réalisée avec des réseaux semblables. Nous allons en particulier montrer les propriétés de compétition spatiale et temporelle qui permettent la différenciation des unités de mouvement. Il est possible d'extraire beaucoup d'autres catégories : ces résultats sont donc présentés à titre d'exemples. Beaucoup de travaux ont été menés dans tous les domaines couverts par les exemples proposés. Nous ne prétendons pas à l'aide de nos résultats faire quantitativement mieux, mais nous voulons montrer qu'une méthode unique peut couvrir tous ces domaines et que la notion de codage par position permet à partir d'une manière unique, de traiter, stocker et retrouver n'importe quel type d'information sans qu'une modélisation préalable ne soit nécessaire.

V.2.1 Architecture des réseaux

La figure V.1, et en particulier l'encadré supérieur gauche montre comment les réseaux utilisés dans les exemples sont connectés. Les cartes sont des tableaux d'unités de traitement. Les réseaux sont organisés de façon rétinotopique ou sonotopique. La majorité des cartes contient le même nombre d'unités et a les mêmes dimensions. Une unité à une position (x,y) donnée dans sa carte observe des ensembles d'unités centrés autour ce cette même coordonnée dans leurs cartes afférentes respectives. Il peut arriver que pour des raisons d'efficacité une réduction soit appliquée entre deux cartes, dans ce cas la région observée dépend du rapport de réduction. Les propriétés d'apprentissage de catégorie que nous souhaitons mettre en évidence sont indépendantes de l'échelle d'observation. À ce niveau de traitement, les différentes échelles n'interagissent pas entre elles et sont indépendantes.

Les dimensions des vecteurs de poids (les vecteurs sont rangés en 2 dimensions) sont choisies dans ces exemples pour contenir les motifs susceptibles d'être extraits. Ils sont suffisamment grands pour que les motifs extraits soient suffisamment riches. Si la nature des motifs à extraire n'était pas connue a priori il serait nécessaire de recourir à une analyse multiéchelle et ainsi de couvrir des motifs de tailles différentes. Nous nous limitons ici à une seule échelle spatiale.

Comme le montre l'encadré de la figure V.1 nous avons utilisé une connectivité partagée, cela signifie que chaque carte utilise le même vecteur de poids. Par la suite toutes les unités de traitement d'une carte donnée vont extraire la même propriété, mais à des positions différentes. Le fait de partager le vecteur de poids pour une carte confère une propriété d'invariance à la translation à ces mêmes cartes.

NeuSter permet également d'attribuer un vecteur de poids différent à chaque unité de traitement. Nous ne présenterons pas de tels résultats ici.

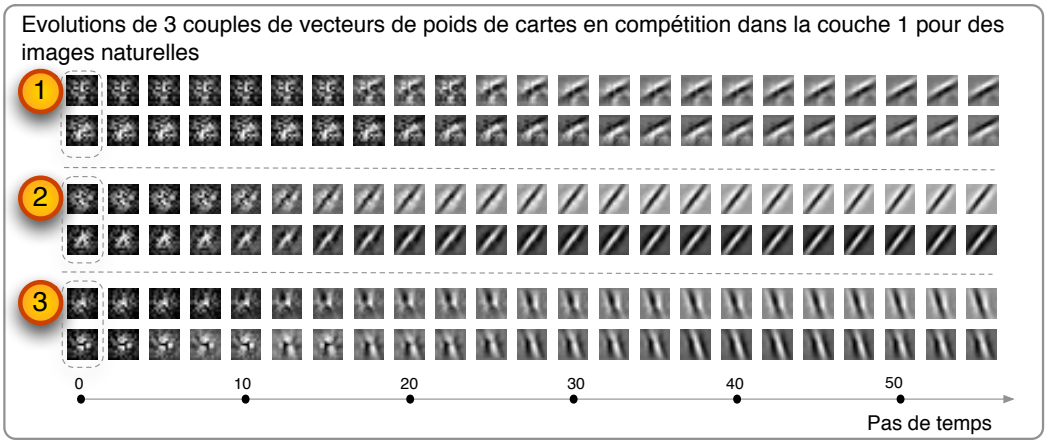


FIG. V.2 – Evolution des vecteurs de poids d'unité en compétitions de la première couche.

V.2.2 Compétitions spatiales et temporelles

Les inhibitions latérales et l'évolution des seuils des unités de traitement sont à l'origine des propriétés de compétition spatiales et temporelles. Des unités en compétition sont des unités appartenant à des cartes différentes de la même couche et situées à la même position dans leur carte respective. Des unités en compétition observent les mêmes données. Les compétitions spatiales correspondent au cas où une unité particulière gagnerait une compétition et empêcherait les autres unités avec lesquelles elle est en compétition d'apprendre le motif d'entrée qui est la cause de leur activation. Dans le cas des cartes à vecteurs de poids partagés, nous empêchons également les unités voisines des unités en compétition d'apprendre leur motif d'entrée. De cette façon, les unités vont se spécialiser chacune dans l'extraction d'une partie différente de l'information. Malgré le fait qu'elles ne puissent pas apprendre, les unités qui perdent leur compétition transmettent leur état d'activation $\beta_i(t)$. Une même coordonnée peut donc s'activer dans différentes cartes. Les différentes cartes réalisent une décomposition de l'information en motifs élémentaires.

Les compétitions temporelles correspondent au cas où les motifs responsables de la spécialisation des unités ne sont pas présentés simultanément à différentes positions, mais séquentiellement. Les vecteurs de poids et les seuils des unités de traitement sont initialement distribués aléatoirement (dans le cas de cartes à vecteurs de poids partagé, il n'y a qu'un vecteur de poids et qu'un seuil par carte). Si aucune unité ne s'active alors qu'il y a un signal d'entrée, alors les seuils de toutes les unités en compétition descendent. Si une unité s'active alors que

les autres ne s'activent pas deux choses vont se produire : 1- l'unité vainqueur va se spécialiser dans le motif qui est la cause de cette activation et son seuil va remonter de telle sorte qu'elle s'active de moins en moins pour les motifs autres que celui pour lequel elle s'est spécialisée ; 2- lorsqu'une unité gagne, les seuils de ses compétitrices ne descendent pas pour éviter une redondance d'information extraite. Nous avons proposé dans le modèle que les seuils remontent plus rapidement qu'ils ne descendent. Ceci évite que des unités spécialisées ne se remettent en jeu trop tôt. La règle en deux points énoncés ci-dessus traite également de la remise en jeu. Lorsque toutes les unités sont spécialisées, qu'il y a des entrées et qu'aucune unité ne s'active, alors les seuils redescendent. Avec un seuil plus bas les unités peuvent se réactiver et changer leur motif favori.

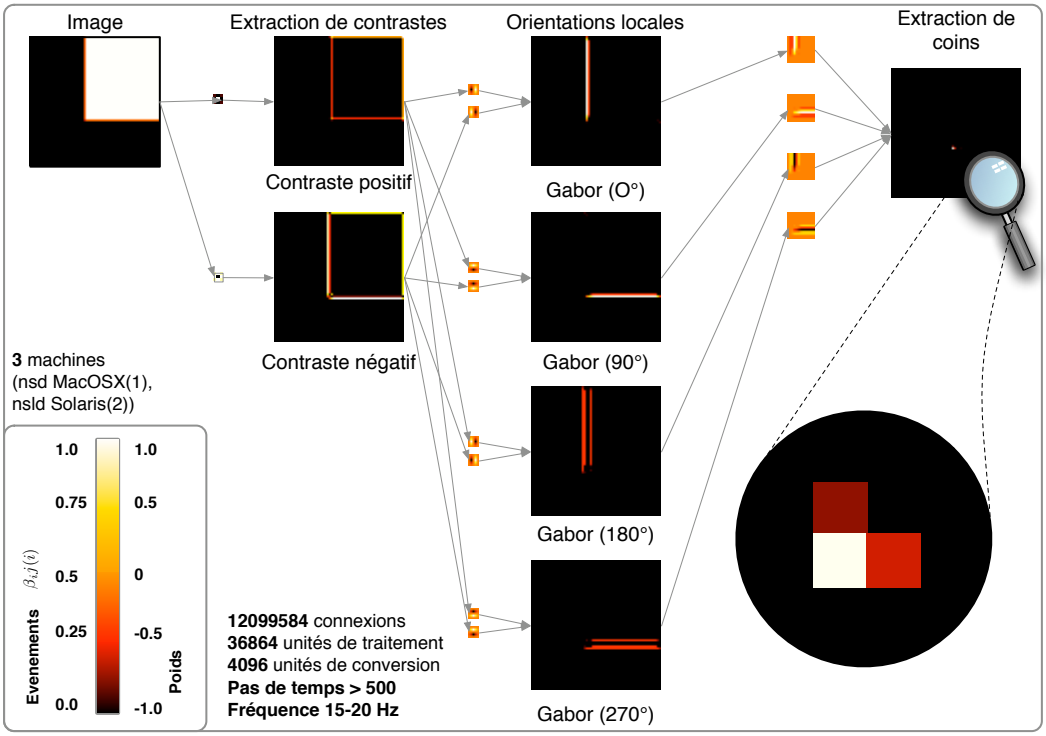


FIG. V.3 – Catégorisation et localisation de coins.

V.2.3 Catégorisation de motifs visuels statiques

Les entrées d'un système utilisant un codage par position doivent elles-mêmes respecter un codage par position. C'est le cas des images acquises par une caméra ou produites par un logiciel de traitement d'image. Chaque pixel peut prendre un ensemble fini de valeurs, et la

position relative des pixels forme des motifs susceptibles d'être catégorisés par le système.

La première couche du système extrait les contrastes positifs et négatifs de l'image d'entrée. Ce pré-traitement permet d'éliminer les régions uniformes et de ne traiter que les variations spatiales des données brutes. On retrouve ce pré-traitement dans tous les exemples de ce chapitre. Il existe une justification plus profonde au fait de réaliser ce pré-traitement. Nous développerons cette justification dans la section V.2.6.

La figure V.1 montre comment les unités de la première couche se spécialisent dans l'extraction de traits et de bords au cours du temps lorsque le système observe des images naturelles. La spécialisation des unités de la première couche est extrêmement rapide (Voir figure V.2). Les vecteurs de poids des unités de traitement convergent vers des extracteurs de bords et de traits en quelques dizaines de pas de temps.

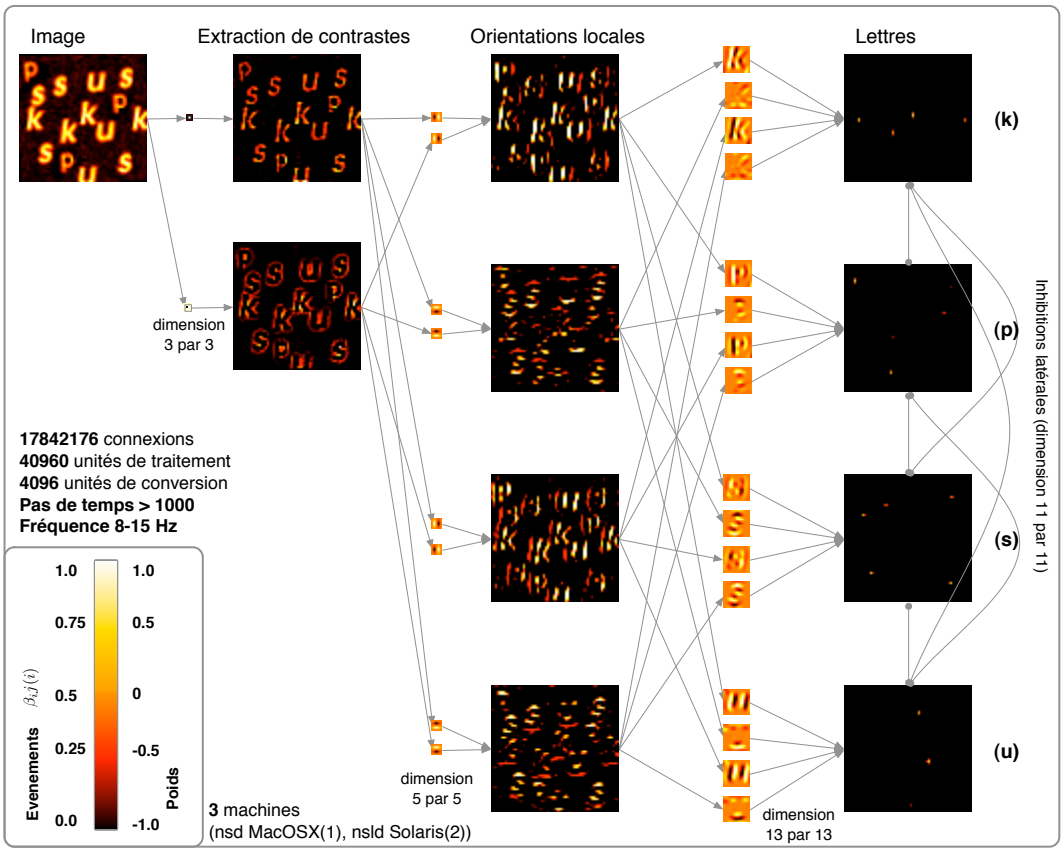


FIG. V.4 – Catégorisation et localisation de lettres.

La figure V.3 illustre l'extraction d'éléments géométriques de second ordre à partir de la seconde couche. Les unités de la seconde couche observent des unités dont l'activation corres-

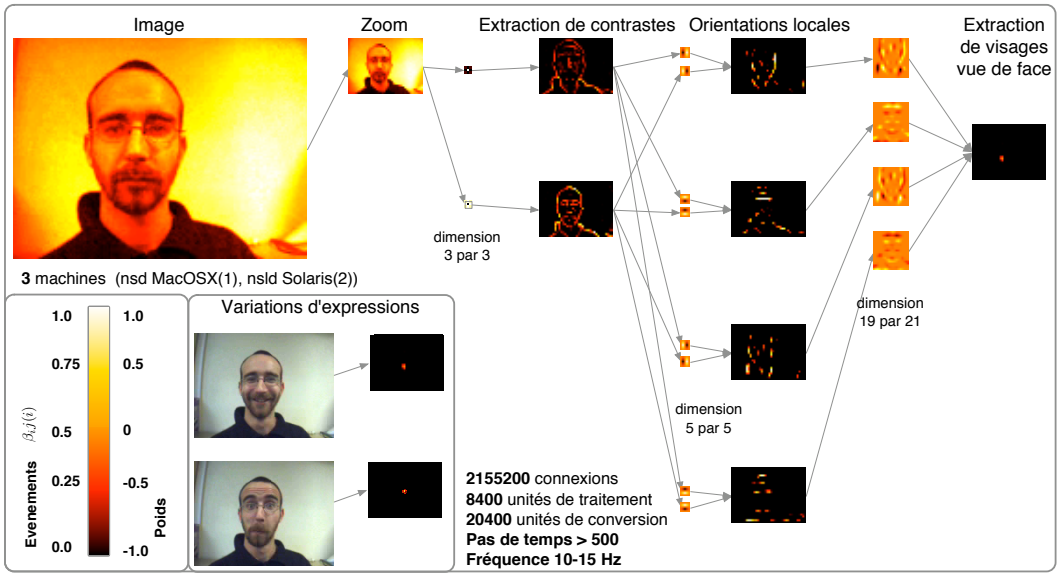


FIG. V.5 – Catégorisation et localisation de visages vus de face.

pond à des éléments géométriques de premier ordre (traits et bords). La cause de l'activation d'une unité dépend de deux orientations locales opposées dans son champ récepteur. Ce type de motifs caractérise des coins ou des terminaisons de traits. Il n'y a rien de spécifique à l'extraction de ces éléments géométriques de second ordre dans le modèle que nous avons proposé. Les unités utilisent toutes le même ensemble fini d'algorithmes.

Les motifs ainsi extraits sont, selon leur nature, plus ou moins résistants aux variations d'angles et d'échelle. Des coins sont invariants aux changements d'échelle, mais ne sont plus détectés au-delà de $\pm 15^\circ$. Pour couvrir l'ensemble des cas, il faut suffisamment d'unités en compétition. Analyser les images en utilisant la statistique d'apparition des éléments de second ordre permet de simplifier considérablement la description géométrique des motifs observés. C'est le rôle de la troisième couche. L'extraction d'un segment à partir de la troisième couche est considérablement simplifiée. Par définition un segment est composé de deux terminaisons de traits opposées et voisines. La présence d'éléments géométriques de second ordre assure la présence d'éléments géométriques de premier ordre. Détecter une terminaison de trait assure qu'un trait est présent. Si une région contient deux et seulement deux terminaisons de trait à des positions différentes, cela signifie que le système observe un segment unique. Au-delà de deux terminaisons, il y a nécessairement plus qu'un simple segment dans l'image observé. D'une manière générale, les catégories visuelles extraites à partir de la troisième couche sont plus stables. Elles se construisent en fonction de la probabilité de présence d'éléments géométriques de second ordre. Cette probabilité de présence est apprise dans les vecteurs de poids des unités.

Plus on monte dans les couches, plus l'information extraite est stable.

La figure V.4 illustre la notion de compétition spatiale. Nous ne présentons dans cet exemple qu'une image contenant 4 motifs représentant des lettres (k,p,s,u). Le réseau possède 2 couches, la seconde couche est constituée d'unités dont les dimensions des vecteurs de poids contiennent potentiellement chaque motif dans leur intégralité. Les unités de la seconde couche se spécialisent très rapidement dans l'extraction de motifs présentés. Les variations d'échelles acceptées sont de l'ordre de $\pm 10\%$ et les variations d'angle de l'ordre de $\pm 10^\circ$. Sur un cluster de 3 machines, la convergence est terminée après 30 secondes de simulation et le système catégorise et localise les 15 cibles entre 8 et 15 *Hz*. Ce ne sont pas les lettres elles-mêmes qui sont reconnues dans ce cas, mais simplement le motif constitué des lettres de cette fonte particulière avec cette orientation et cette taille particulière. On peut voir ici une illustration de la notion de codage par position. C'est le fait qu'une unité s'active dans une carte particulière qui peut être interprété comme un sens, ici le fait d'avoir une forme de lettre.

La figure V.5 illustre la même propriété que la catégorisation et la localisation de visage. Le modèle de visage pour lequel se spécialisent les unités de la seconde couche accepte de fortes variations de taille et d'angle, ainsi que des variations d'expressions dans le visage. Cette robustesse est due à une réduction de l'image d'entrée et au fait que la loi d'apprentissage produit un vecteur de poids moyen contenant les composantes principales de la forme du visage. On peut en particulier remarquer la barre des yeux, les bords du visage et le col dans les motifs recherchés par les unités de traitement de la seconde couche. Dans cet exemple, le réseau apprend une forme, et la position où les unités s'activent dans la carte permet de déduire la position du motif extrait dans l'image d'entrée. Cette propriété de localisation a été utilisée pour réaliser un suivi de cible à l'aide d'un module pan/tilt sur lequel était placée une caméra. Cette manipulation n'utilise pas les propriétés déduites de l'analyse sur la façon dont un système de ce type doit agir. Il s'agit d'une application simple d'apprentissage et de suivi de cible. Le module pan/tilt reçoit une commande calculée en fonction de la position de l'objet cible et déplace la caméra de façon à ce que la position de l'objet observé se rapproche du centre de l'image. Si la cible change peu de taille, le système constitué de la caméra et du module pan/tilt parvient à suivre la cible.

L'extraction d'information repose pour une grande part sur les vecteurs de poids des unités de traitement. On observe que les éléments caractéristiques sont représentés par des coefficients positifs ou négatifs. Les coefficients positifs constituent les éléments qui favorisent l'activation de l'unité au contraire des éléments négatifs qui défavorisent son activation. Une grande partie des coefficients tend vers 0. Si une unité active est observée à travers un coefficient proche de 0 elle ne joue aucun rôle dans l'activation de l'unité observante. On peut assimiler cette propriété à une transparence. Plusieurs informations peuvent être superposées et détectées à une même position. Si une unité trouve son motif favori dans une superposition de motifs, elle s'activera en vertu des différents 0 qui peuplent son vecteur de poids. L'extraction d'informations est robuste à des variations d'arrière-plan, ou à des motifs qui viendraient partiellement cacher la cible.

Les propriétés déduites de l'analyse montrent que l'extraction des catégories de haut niveau ne doit pas être effectuée à toutes les positions. Si c'était le cas, les unités véhiculeraient

en s'activant une double information sur la nature et le lieu de la propriété observée. Dans ce cas, l'information de lieu ne serait pas exploitable pour que le système puisse généraliser cette notion. Plus on s'éloigne des entrées, plus les unités doivent extraire une information indépendante de la position. La position doit être donnée par un système parallèle.

V.2.4 Extraction de motifs visuels dynamiques

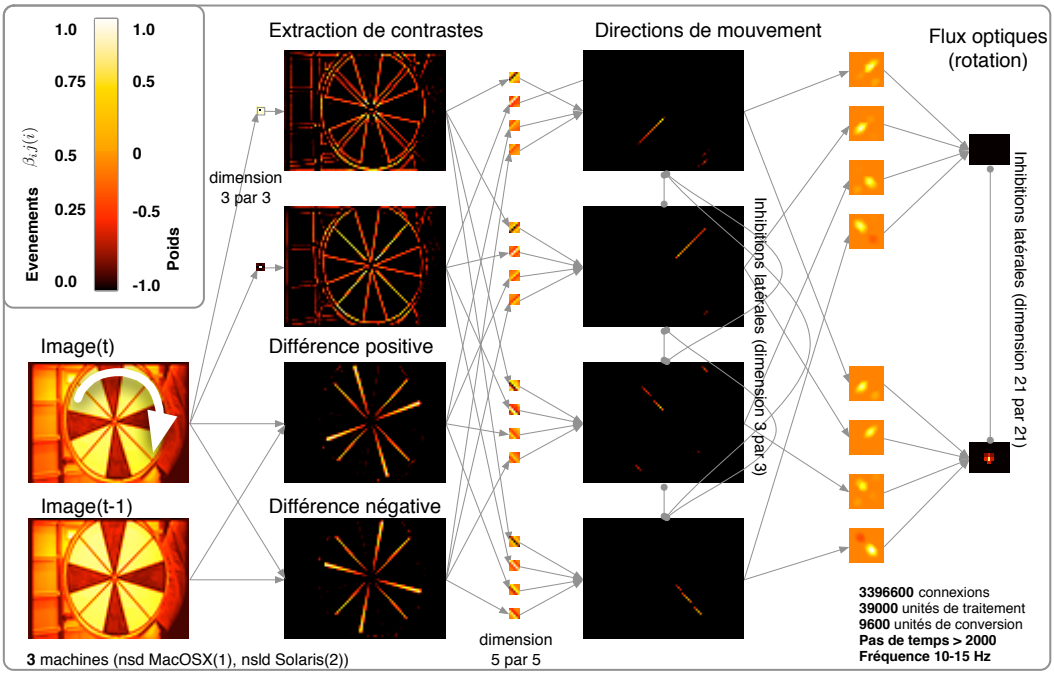


FIG. V.6 – Catégorisation de mouvements apparents et flux optiques.

Nous avons également utilisé ce même modèle pour catégoriser des directions de mouvement et des flux optiques. Cette fois les images d'entrées doivent dépendre du temps. Nous utilisons 2 images temporellement voisines (t et $t-1$). La couche de prétraitement est cette fois constituée de 4 cartes, deux cartes vont extraire de l'information statique de contraste, et 2 autres vont extraire de l'information dynamique en proposant la différence des deux images consécutives sous deux formes ($Image(t) - Image(t-1)$ et $Image(t-1) - Image(t)$).

Les unités de la première couche cherchent alors à extraire des motifs stables à partir de ces 4 entrées. Très rapidement les unités se spécialisent dans l'extraction de direction de mouvement (Voir figure V.6). L'image observée contient une roue en rotation dans le sens horaire puis antihoraire. Les unités de la première couche se spécialisent dans la direction de mouvement formé par les 4 diagonales. On peut observer les rayons de la roue décomposés à travers la

première couche. Là encore, la carte où apparaît le rayon véhicule la sémantique de la décharge des unités.

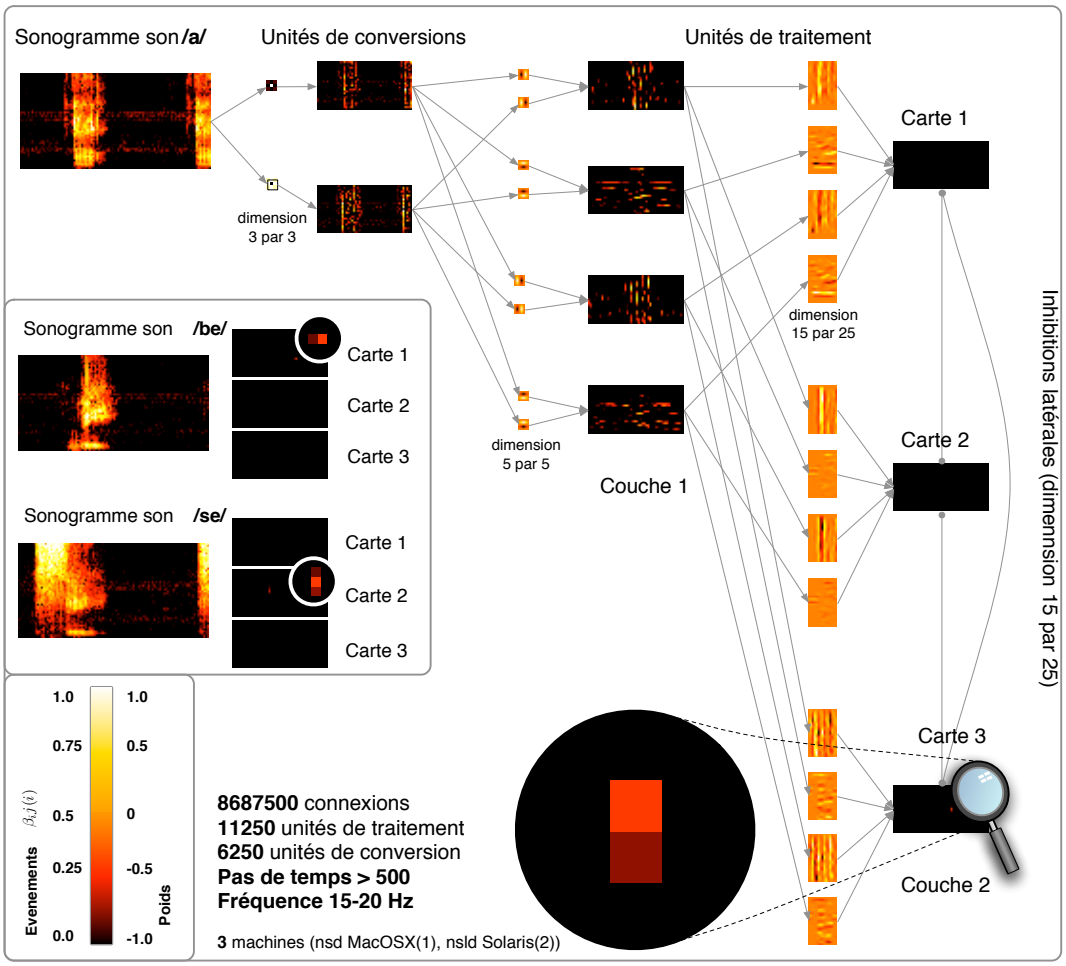


FIG. V.7 – Catégorisation de phonèmes.

La seconde couche se spécialise dans l'extraction de rotations dans le sens horaire et antihoraire. Cette spécialisation se produit après une compétition entre les deux cartes de la seconde couche.

Ce type d'approche est donc suffisamment général pour catégoriser des formes statiques et dynamiques.

V.2.5 Extraction de motifs sonores

Nous avons voulu aller plus loin en testant un tel système sur des sonagrammes. Le son dans ce cas est présenté sous la forme d'une image et respecte la propriété de codage par position.

La figure V.7 montre que les unités de la seconde couche d'un tel réseau se spécialisent rapidement dans l'extraction de sons. Les entrés sont constituées de 3 phonèmes : **/a/**, **/be/**, et **/se/**. Ces phonèmes ont été produits par un synthétiseur vocal (voix féminine du logiciel Speechissimo de la société Elan, PowerPC@700MHz). Le son a été produit et acquis par les haut-parleurs et le micro d'une même machine. Le traitement a été effectué en temps réel.

On remarque que les unités de la seconde couche se spécialisent rapidement dans l'extraction des phonèmes présentés. La propriété de superposition des motifs permise par la présence de poids proche de 0 dans les vecteurs de poids confère à ce système une très bonne résistance au bruit.

V.2.6 Conclusion et perspective

Nous avons montré les premiers résultats du système de représentation ouvert que nous souhaitons réaliser. Ces expérimentations ont été réalisées à l'aide de NeuSter, logiciel développé au cours de cette thèse. Ces résultats illustrent en particulier la capacité de spécialisation des unités de traitement en fonction de leur position dans le réseau. L'extraction de ces motifs est totalement non supervisée.

Nous avons choisi de ne pas nous appuyer sur les notions inspirées par les neurosciences pour justifier notre approche : en effet, nous considérons que la notion de système ouvert va au-delà de cette discipline, et qu'il existe un ensemble de concepts que l'on peut retrouver à la fois dans l'anatomie et le fonctionnement du cerveau et qui peuvent également être mis en oeuvre dans une machine. Le cerveau et la machine associés à un logiciel particulier pourraient alors avoir des propriétés communes sans qu'une approche ait à se réclamer de l'autre.

Les prochaines expérimentations vont porter sur l'extraction de motifs indépendamment du point de vue ou de la taille. Ces propriétés reposent dans le modèle sur la capacité d'associations temporelles. D'autres expérimentations vont porter sur l'association de catégories visuelles et sonores. Notre approche nécessite la mise au point de réseaux fonctionnels. Le darwinisme neural proposé par Edelman semble être une piste intéressante pour le développement de réseaux adaptés à des types particuliers de représentations (forme, couleur, mouvement) [Edelman 82],[Edelman 87]. Ce type d'approche consiste à générer un grand nombre de réseaux, en faisant varier les différents paramètres, et à sélectionner ceux qui se montrent les plus adaptés à une fonction particulière.

Un système attentionnel est également en cours de développement. Il doit permettre au système de choisir les prochaines observations à effectuer et doit proposer la position relative de deux observations sous la forme de l'état proprioceptif d'un module pan/tilt. Les premières actions vont consister à maîtriser la direction du regard en fonction de l'intérêt des objets observés et d'apprendre à associer la production de mots avec les propriétés observées par essai erreur. Certaines de ces expérimentations devraient être réalisées au moment de la soutenance.

Chapitre VI

Conclusion et perspectives

Il est difficile de définir la robotique en une phrase. Cette science regroupe de nombreuses approches dont les objectifs peuvent être très différents. On peut distinguer deux classes principales d'approches. La première consiste à construire des machines ayant des degrés divers d'autonomie qui apportent un ensemble de fonctionnalités ou de services à l'homme. La seconde considère le robot comme un moyen de valider des modèles issus d'autres disciplines. C'est en particulier le cas des modèles évolutionnistes ou de modèles issus des neurosciences computationnelles. La première approche considère le robot et ses fonctionnalités comme une fin alors que la seconde le considère comme un moyen d'étude. Ces deux approches peuvent être liées. Quelles que soient les approches, les logiciels développés pour faire fonctionner les robots sont issus d'un travail de modélisation, et sont spécifiques à l'ensemble des tâches dont on souhaite doter la machine. Les modèles portent à la fois sur le système de représentation de l'environnement et sur les fonctionnalités dont doit disposer la machine. Les fonctionnalités sont toujours construites à partir des représentations dont disposent les robots. L'étape fondamentale de toute conception logicielle est basée sur la modélisation d'un problème qu'on souhaite résoudre, en particulier les représentations dont doit disposer la machine pour agir. Il y a autant de systèmes de représentation que de problèmes à résoudre, il est de plus très complexe de faire cohabiter des systèmes de représentation issus d'approches différentes.

Cette thèse propose une analyse, une modélisation et une mise en oeuvre d'un système d'apprentissage ouvert de représentations et de fonctionnalités en robotique. Nous avons défini la notion d'ouverture comme l'absence de contraintes explicites relatives à l'objectif du système. Plutôt que de spécifier les représentations et les fonctionnalités du système pendant sa phase de conception, nous souhaitons donner à la machine la capacité d'acquérir seule de l'informa-

tion de son environnement, de la traiter, la stocker, et la réutiliser. Nous souhaitons également qu'elle soit en mesure de construire ses propres fonctionnalités à travers son expérience en se basant sur les représentations qu'elle aura acquises.

L'analyse du problème que constitue la construction d'un système de représentation ouvert nous a conduit à une analogie intéressante. Représenter des catégories et des quantités constituent deux problèmes semblables. Au cours de l'histoire, de très nombreux systèmes de numérations ont été proposés et utilisés. Le système de numération romain est certainement le plus connu des systèmes de numération archaïque. Ecrire 2^{64} en utilisant un tel système de numération oblige à créer de nouveaux symboles. Additionner *MMCI* et *CIX* n'est pas possible autrement qu'en comptant sur ses doigts ou en ayant appris par coeur l'ensemble des opérations possibles. Le problème principal posé par cette classe de systèmes de numération est dû au fait que le nombre est directement codé dans les chiffres. Après plusieurs siècles de tâtonnement, une solution extrêmement efficace et élégante à été proposée. Le système de numération par position peut être considéré comme le premier système ouvert artificiel. Il utilise un ensemble fini de chiffres et les règles arithmétiques ne portent que sur des chiffres et jamais sur les nombres eux-mêmes. L'infinité des quantités exprimables est alors transférée dans l'infinité des positions où l'on peut écrire les chiffres. Le sens, c'est à dire, la valeur du nombre est abstraite. Le sens des chiffres, c'est à dire la quantité qu'ils expriment, est codé dans leur position. Le zéro permet en particulier de signifier l'absence d'une quantité.

Les systèmes de représentation utilisés en robotique aujourd'hui sont issus d'un processus de modélisation. Ces représentations sont des ensembles de symboles qui réfèrent explicitement à des propriétés du monde. Coder les représentations qu'un robot a de son environnement est analogue au codage des quantités mesurables. Considérer des symboles codant explicitement un sens dans le cas d'un système de représentation est équivalent aux systèmes de numération primitif. Il y a une infinité de représentations qu'il faut artificiellement construire, et une infinité d'algorithmes portant sur ces représentations.

Nous proposons dans cette thèse d'utiliser un tel système pour bâtir un système de représentation ouvert. Nous proposons un système général issu d'un raisonnement d'analyse dont l'une des caractéristiques principales est de reprendre l'idée des systèmes de codage par position. L'infinité des sens qui caractérisent les catégories observables est rejetée dans les lieux ou ces catégories sont codées. La question de la sémantique ne se pose jamais explicitement. Ce type d'approche est également caractérisable par le fait qu'un nombre fini d'algorithmes et d'états suffit à traiter tous les cas. Cette propriété est à mettre en parallèle avec le fait qu'un système de numération par position n'utilise qu'un nombre fini de chiffres et que les lois arithmétiques sont en nombre fini, car elle ne porte que sur les chiffres et plus sur les nombres, c'est à dire le sens. Les algorithmes de propagation d'information et d'apprentissage relatifs à une telle approche doivent toujours être locaux. Un tel système de représentation ouvert permet de représenter n'importe quelle catégorie pour peu que le système dispose de suffisamment d'unités de traitement.

L'analyse nous conduit à la conclusion qu'un système de représentation qui soit en mesure de généraliser des situations doit être distribué et doit rendre compte de plusieurs catégories simultanément. Il doit proposer une décomposition des informations perceptives. La notion de

représentation distribuée est à l'origine de la capacité de généralisation. C'est en cherchant à réactiver les unités codant pour des parties stables des représentations que le système doit guider la production de ses actions. Face à une situation inconnue, il pourra trouver un point commun à une situation déjà rencontrée et maîtrisée et agir de façon appropriée.

L'analyse du problème a permis d'exhiber un ensemble de propriétés nécessaires ou suffisantes à la conception d'un tel système. Le système de représentation ouvert de l'environnement constitue la base du système de fonctionnalité ouvert. L'ensemble des propriétés exhibées ne conduit pas à un modèle unique, mais à une classe de systèmes équivalents. Nous avons construit un modèle particulier à la suite de l'analyse. Ce modèle prend en compte l'ensemble des propriétés exhibées à l'exception de la propriété de superposabilité des états.

Un simulateur parallèle a été développé pour mettre en oeuvre cette classe de système. Ce simulateur appelé NeuSter permet de répartir la charge des calculs nécessaires au fonctionnement d'un tel système sur un cluster hétérogène de machines POSIX. NeuSter est hautement configurable, et permet de garder en temps réel la main sur l'ensemble des paramètres qui constituent un réseau. Ce simulateur à été utilisé pour nos premières expérimentations. NeuSter est rapide et peut mettre à jour plusieurs millions de connexions par seconde sur un seul noeud de calcul. Il permet de simuler des réseaux de taille arbitrairement grande et en temps réel pour peu que le cluster utilisé dispose de suffisamment de machines.

Les résultats que nous présentons dans cette thèse visent à valider la notion de codage de représentation par position associée à un apprentissage non supervisé. Nous montrons à titre d'exemple qu'il est possible d'extraire des images naturelles des orientations locales sur la première couche d'un réseau. La seconde couche permet d'extraire des éléments géométriques de second ordre tels que des coins ou des terminaisons de trait. En utilisant des vecteurs de poids plus grands, il est également possible d'apprendre à extraire des visages vu de face dès la seconde couche. Pour aller plus loin nous avons appliqué cette méthode de codage par positon à l'apprentissage de directions de mouvement apparent en utilisant des séquences de paires d'images spatialement proches. Il est possible d'apprendre des directions de mouvement sur la première couche et des flux optiques sur la seconde couche (rotation horaire et antihoraire, expansion, contraction, cisaillement). Enfin, nous avons montré que ce type de codage permettait aussi d'extraire des phonèmes à partir de sonagrammes.

Les travaux futurs devront porter sur les méthodes de construction de ces réseaux. La principale piste actuelle consiste à considérer que le système distribué de catégories doit extraire des contrastes sémantiques, c'est à dire chercher dans des régions de l'espace proches l'activation d'unités codant pour des sens opposés. La première étape de traitement consiste à extraire des contrastes, c'est-à-dire à chercher dans des régions voisines la présence de lumière et d'obscurité. La seconde étape consiste à trouver dans une même région des alignements de contrastes positifs et de contrastes négatifs. La troisième étape consiste à trouver des verticales et des horizontales voisines. On voit se dégager une notion de contraste sémantique généralisé. Il semble exister une solution qui permette de réaliser des connexions par paire. En faisant en sorte que des unités en compétition respectent une topologie de connexion circulaire et en faisant en sorte que les voisins topologiques des gagnants des compétitions adaptent leurs poids simultanément avec les vainqueurs (voir approche SOM de Kohonen), il doit être possible

d'obtenir des unités diamétralement opposées au sens du voisinage topologique circulaire et codant pour des propriétés sémantiquement opposées. De cette façon, des unités observant des groupes d'unités diamétralement opposées serait en mesure d'extraire un contraste sémantique sans qu'il soit nécessaire pour le concepteur du système de se préoccuper quelles unités codent pour quels sens.

Les perspectives de ce type d'approches sont importantes. Il n'est plus nécessaire de modéliser au préalable les représentations et l'ensemble des tâches que l'on souhaite faire réaliser à la machine. Elle apprend seule en utilisant son expérience à construire ses représentations et ses fonctionnalités. Concevoir un tel système revient à décrire un réseau.

Références bibliographiques

[Alami 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy*. International Journal of Robotics Research, vol. 17, no. 4, pages 315–337, 1998.

[Arsenio 03] A. Arsenio, P. Fitzpatrick, C.C. Kemp & G. Metta. *The Whole World in Your Hand : Active and Interactive Segmentation*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Barlow 72] H.B. Barlow. *Single units and sensation*. Perception, vol. 1, pages 371–394, 1972. grandmother cell.

[Bell 97] A.J. Bell & T.J. Sejnowski. *The 'Independent Components' of Natural Scenes are Edge Filters*. Vision Research, vol. 37, no. 23, pages 3327–3338, 1997.

[Bickhard 95] M.H. Bickhard & L.G. Terveen. Foundational issues in artificial intelligence and cognitive science : Impasse and solution. Elsevier Science, 1995.

[Bouchired 98] S. Bouchired, M. Ibnkahla & W. Paquier. *A combined LMS-SOM receiver for rapidly fading nonlinear channels*. In EUSIPCO, Rhodes, Greece, 1998.

[Breidegard 03] B. Breidegard & C. Balkenius. *Speech Development by Imitation*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Brooks 86] R. A. Brooks. *A Robust Layered Control System For a Mobile Robot*. IEEE Journal of Robotics and Automation, vol. RA-2, no. 1, pages 14–23, 1986.

[Brooks 90] R. Brooks. *Elephants Don't Play Chess*. Robotics and Autonomous Systems, vol. 6, no. 1&2, pages 3–15, 1990.

[Brooks 91a] R. Brooks. *Challenges for Complete Creature Architectures*. In Jean-Arcady Meyer & Stewart W. Wilson, editeurs, From animals to animats, pages 434–443, 1991.

[Brooks 91b] R. Brooks. *How to build complete creatures rather than isolated cognitive simulators*, 1991.

[Brooks 91c] R. Brooks. *Intelligence Without Reason*. In John Myopoulos & Ray Reiter, editeurs, Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann publishers Inc. : San Mateo, CA, USA.

[Brooks 91d] R. Brooks. *The role of learning in autonomous robots*. In Proceedings of the Fourth Annual Workshop on Computational Learning Theory, pages 5–10, San Mateo, CA, 1991. Morgan Kaufmann.

[Brooks 91e] Rodney A. Brooks. *Intelligence without Representation*. Artificial Intelligence, vol. 46, pages 139–159, 1991.

[Brooks 92] R. Brooks. *Artificial Life and Real Robots*. In European Conference on Artificial Life, pages 3–10, 1992.

[Brooks 94] R. A. Brooks. *Coherent behavior from many adaptive processes*. In From Animals to Animats 3 : Proceedings of The Third International Conference on Simulation of Adaptive Behavior, pages 22–29. MIT Press/Bradford Books, Cambridge, MA, 1994.

[Brooks 98] R. A. Brooks, C. Breazeal, R. Irie, C. C. Kemp, M. Marjanovic, B. Scassellati & M. M. Williamson. *Alternative Essences of Intelligence*. In Proceedings of the 15th AAAI, AAAI Press, pages 961–968, 1998.

[Delorme 99] A. Delorme, R. Van Rullen, J. Gautrais & S..J. Thorpe. *SpikeNET : a simulator for modeling large networks of integrate and fire neurones*. Neurocomputing, vol. 26-27, pages 989–996, 1999.

[Delorme 03] A. Delorme & S. J Thorpe. *SpikeNET : an event-driven simulation package for modelling large networks of spiking neurons*. Network : Computation in Neural Systems, vol. 14, pages 613–627, November 2003.

[Dreyfus 79] H. L. Dreyfus. What computers can't do : The limits of artificial intelligence. New York : Harper & Row, 1979.

[Edelman 82] G.M. Edelman. *Selective Networks capable of Representative Transformations, Limited Generalizations, and Associative Memory*. Proc. Nat. Acad. Sci., vol. 79, pages 2091–2095, 1982.
Description of the "Darwin" machine.

[Edelman 87] G.M. Edelman. Neural darwinism : The theory of neuronal group selection. Basic Books, New York, 1987.
Presentation of Edelman's theory that high-level functions compete for neuronal support.

[Engel 91a] A. K. Engel, P. König, A. K. Kreiter, & W. Singer. *Interhemispheric synchronization of oscillatory neuronal responses in cat visual cortex*. Science, no. 252, pages 1177–1179, 1991.

[Engel 91b] A. K. Engel, P. König, A. K. Kreiter, & W. Singer. *Synchronization of oscillatory neuronal responses between striate and extrastriate visual cortical*

areas of the cat. Proceedings of the National Academy of Sciences USA, no. 88, pages 6048–6052, 1991.

[E. Oja 82] E. Oja. *A simplified neuron model as a principal component analyser*. Journal of Mathematical Biology, vol. 15, pages 267–273, 1982.

[Feigenbaum 63] E.A. Feigenbaum & J. Feldman. Computers and thought. McGraw-Hill, Cambridge, 1963.

[Floreano 94] D. Floreano & F. Mondada. *Automatic Creation of an Autonomous Agent : Genetic Evolution of a Neural-Network Driven Robot*. In Proceedings of the Conference on Simulation of Adaptive Behavior, 1994.

[Gaskett 99] C. Gaskett, D. Wettergreen & A. Zelinsky. *Q-Learning in Continuous State and Action Spaces*. In 12th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, December 1999. Lecture Notes in Computer Science, Springer-Verlag.

[Gaskett 00] C. Gaskett, L. Fletcher & A. Zelinsky. *Reinforcement Learning for a Vision Based Mobile Robot*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2000), Takamatsu, Japan, November 2000.

[Gaskett 03] C. Gaskett, P. Brown, G. Cheng & A. Zelinsky. *Learning Implicit Models during Target Pursuit*. In IEEE International Conference on Robotics and Automation (ICRA2003), Taiwan, May 2003.

[Gautrais 98] J. Gautrais & S..J. Thorpe. *Rate coding versus temporal order coding : a theoritical approach*. Neurocomputing, vol. 48, pages 57–65, 1998.

[Gergely 03] G. Gergely. *What should a robot learn from an infant*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Gerstner 99] W. Gerstner, R. Kempter, J. van Hemmen & H. Wagner. *Hebbian learning of pulse timing in the barn owl auditory system*, 1999.

[Gray 89] C. M. Gray, P. König, A. K. Engel, & W. Singer. *Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties*. Nature, no. 338, pages 334–337, 1989.

[Grossberg 91a] S. Grossberg & D. Somers. *Synchronized oscillations during cooperative feature linking in a cortical model of visual perception*. Neural Networks, no. 4, pages 453–466, 1991.

[Grossberg 91b] S. Grossberg & L. Wyse. *A neural network architecture for figure-ground separation of connected scenic figures*. Neural Networks, no. 4, pages 723–742, 1991.

[Grupen 03] R. A. Grupen. *A developmental Organization for Robot Behavior*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Guillot 02] A. Guillot. Introduction : Histoire des systèmes cognitifs artificiels., approche dynamique de la cognition artificielle. A. Guillot et E. Daucé (Eds), Hermès, 2002.

[Hallam 02] J. Hallam, D. Floreano, G. Hayes & J.-A. Meyer. From animals to animats 7. proceedings of the 7th international conference on adaptive behavior. MIT Press, Cambridge, MA, 2002.

[Harnad 90] S. Harnad. *The symbol grounding problem*, 1990.

[Haykin 94] Simon Haykin. NEURAL NETWORKS A Comprehensive Foundation. Macmillan College Publishing Company, Inc., 866 Third Avenue, New York, New York 10022, 1994.

[Hebb 49] D. O. Hebb. The organization of behavior : A neuropsychological theory. New York : Wiley, 1949.

[Hines 89] M.L Hines. *A program for simulation of nerve equations with branching geometries*. Int. J. Bio-Med, vol. 24, pages 55–68, 1989.

[Hines 01] M.L Hines & Carnevale. *NEURON : a tool for neuroscientists*. Int. J. Bio-Med, vol. 7, pages 123–135, 2001.

[Hodgkin 52] A.L. Hodgkin & A.F. Huxley. *Propagation of Electrical Signals Along Giant Nerve Fibres*. Proceedings of the Royal Society of London. Series B, Biological Sciences, vol. 140, no. 899, pages 177–183, October 1952.

[Holland 75] J.H. Holland. Adaptation in natural artificial systems. University of Michigan Press, Ann Arbor, 1975.

[Hopfield 82] J.J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. In Proceedings of the National Academy of Sciences, volume 79, pages 2554–2558. Washington : The Academy, 1982.

[Kaelbling 96] L. Pack Kaelbling, M. L. Littman & A. P. Moore. *Reinforcement Learning : A Survey*. Journal of Artificial Intelligence Research, vol. 4, pages 237–285, 1996.

[Kandel 91] E. R. Kandel, J. H. Schwartz, & T. M. Jessell. Principles of Neural Science 3rd edition. Elsevier, New York, 1991.

[Kaplan 03] F. Kaplan & P.Y. Oudeyer. *Motivational Principles for Visual Know-how Development*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Karda 87] C.A. Karda & W.J. Freeman. *How brains make chaos in order to make sense of the world*. Behavioral and Brain Sciences, vol. 10, pages 161–195, 1987.

[Koenig 96] S. Koenig & R. Simmons. *Unsupervised learning of probabilistic models for robot navigation*. In Proc. IEEE Int'l. Conf. on Robotics and Automation, 1996.

[Kohonen 82] T. Kohonen. *Self-Organised Formation of Topologically Correct Feature Maps*. Biological Cybernetics, vol. 43, pages 59–69, 1982.

[Kohonen 95] T. Kohonen. Self-organizing maps, volume 30 of *Series in information sciences*. Springer, Berlin, 1995.

[König 91] P. König & T. B. Schillen. *Stimulus-dependent assembly formation of oscillatory responses : I. Synchronization*. Neural Computation, no. 3, pages 155–166, 1991.

[König 95] P. König, A. K. Engel, P. R. Roelfsema, & W. Singer. *How precise is neuronal synchronization ? Neural Computation*. Neural Computation, no. 7, pages 469–485, 1995.

[Kuipers 94] B. Kuipers. Qualitative reasoning : Modeling and simulation with incomplete knowledge. MIT Press, Cambridge, MA, 1994.

[Lapicque 07] L. Lapicque. *Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation*. J. Physiol. Pathol. Gen., vol. 9, pages 620–635, 1907.

[Levine 85] D.N. Levine, J. Warach & M.J. Farah. *Two visual systems in mental imagery : Dissociation of 'what' and 'where' in imagery disorders due to bilateral posterior cerebral lesions*. Neurology, no. 35, pages 1010–1018, 1985.

[Llinás 93] R. Llinás & U. Ribary. *Coherent 40-Hz oscillation characterizes dream state in humans*. Proceedings of the National Academy of Sciences USA, no. 90, pages 2078–2082, 1993.

[Löwel 92] S. Löwel & W. Singer. *Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity*. Science, no. 255, pages 209–212, 1992.

[Lungarella 03] M. Lungarella & G. Metta. *Beyond Gazing, Pointing, and Reaching*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Maes 90a] P. Maes. *Situated Agents Can Have Goals*. In Patti Maes, editeur, Designing Autonomous Agents, pages 49–70. MIT Press, 1990.

[Maes 90b] P. Maes & R. Brooks. *Learning to Coordinate Behaviors*. In National Conference on Artificial Intelligence, pages 796–802, 1990.

[McCulloch 43] W. S. McCulloch & W. Pitts. *A logical calculus of the ideas immanent in nervous activity*. Bull. Math. Biophysics, vol. 5, pages 115–133, 1943.

[Meyer 91] J.-A. Meyer & A. Guillot. *An Architecture for Autonomy*. In From animals to animats. Proceedings of the First International Conference on the Simulation of Adaptive Behavior, pages 2–14, Cambridge, MA, 1991. MIT Press.

[Meyer 00] J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblat & S. Wilson. From animals to animats 6. proceedings of the 6th intl. conf. on simulation of adaptive behavior. MIT Press, Cambridge, MA, 2000.

[Minsky 68] M.L. Minsky. Semantic information processing. MIT press, Cambridge, 1968.

[Mishkin 82] M. Mishkin, L.G. Ungerleider & K.A. Macko. *Object vision and spatial vision : Two cortical pathways*. Trends in Neurosciences, vol. 6, pages 414–417, 1982.

[Mondada, F. 95] Mondada, F. & Floreano, D. *Evolution of neural control structures : Some experiments on mobile robots*. Robotics and Autonomous Systems, vol. 16, no. 2-4, pages 183–195, december 1995.

[Morisset 02] B. Morisset & M. Ghallab. *Learning how to combine sensory-motor modalities for a robust behavior*. In M. Beetz, J. Hertsberg & M. Ghallab, editeurs, Lecture Notes in Plan-Based Control of Robotic Agents. Springer Verlag, 2002.

[Newell 90] A. Newell. Unified theories of cognition. Harvard University Press, Cambridge, MA, 1990.

[Nolfi 01] S. Nolfi & D. Floreano. Evolutionary robotics. the biology, intelligence, and technology of self-organizing machines. T. Gomi (Ed.), Evolutionary Robotics, AAI Books, MIT Press, Cambridge, MA, 2001.

[Olshausen 00] B. A. Olshausen, P. Sallee & M. S. Lewicki. *Learning Sparse Image Codes using a Wavelet Pyramid Architecture*. In NIPS, pages 887–893, 2000.

[O'Regan 96] J.K. O'Regan, J.A. Rensink & J.J. Clark. *"Mud splashes" render picture changes invisible*. Investigative Ophthalmology & Visual Science, vol. 37 :213, pages 157–165, 1996.

[O'Regan 97] J.K. O'Regan, J.A. Rensink & J.J. Clark. *To see or not to see : the need for attention to perceive changes in scenes*. Psychological Science, vol. 8, no. 5, pages 368–373, 1997.

[Paquier 98] W. Paquier & M. Ibnkahla. *Self-organizing maps for rapidly fading nonlinear channel equalization*. In IEEE, editeur, World Congress on Computational Intelligence, Anchorage, Alaska, USA, 1998.

[Paquier 00a] W. Paquier, A. Delormes & S. Thorpe. *Motion Processing Using One Spike per Neuron*. In Computational Neuroscience Meeting, page 115, Brugge, Belgium, 2000.

[Paquier 00b] W. Paquier, A. Delormes, R. Vanrullen & S. Thorpe. *Détection de mouvements apparents par codage asynchrone*. In Xemes Journées Neurosciences et Sciences de l'ingénieur, Dinard, France, 2000.

[Paquier 02] W. Paquier & R. Chatila. *An Architecture for Robot Learning*. In IAS, pages 575–578, 2002.

[Paquier 03a] W. Paquier & R. Chatila. *Learning New Representations and Goals for Autonomous Robots*. In ICRA, 2003.

[Paquier 03b] W. Paquier, N. Do Huu & R. Chatila. *A Unified Model for Developemental Robotics*. In Epigenetic Robotic, 2003.

[Pearson 92] K. Pearson. The grammar of science. Walter Scott, London, 1892.

[Pribram 74] K. Pribram, Nuwer Marc, Baron & J. Robert. *The holographic hypothesis of memory structure in brain function and perception*. Contemporary Developments in Mathematical Psychology, vol. Vol. II, pages 416–457, 1974.

[Rolls. 01] E.T. Rolls. The brain and emotion. Oxford University Press, Oxford, 2001.

[Rosenblatt 58] F. Rosenblatt. *The Perceptron : A probabilistic model for information storage and organization in the brain*. Psychological Review, vol. 65, pages 386–408, 1958.

[Rosenblatt 60] F. Rosenblatt. *On the convergence of reinforcement procedures in simple perceptrons*. Rapport technique VG-1196-G-4, Cornell Aeronautical Laboratory, Buffalo, NY, 1960.

[Rosenblatt 62] F. Rosenblatt. Principles of neurodynamics. Spartan Books, Washington, DC, 1962.

[Ruf 97] B. Ruf & M. Schmitt. *Unsupervised learning in networks of spiking neurons using temporal coding*. In W. Gerstner, A. Germond, M. Hasler & J. D. Nicoud, editeurs, Artificial Neural Networks—ICANN '97. 7th International Conference on Artificial Neural Networks Proceedings, pages 361–6. Springer-Verlag, Berlin, Germany, 1997.

[Ruf 98] B. Ruf & M. Schmitt. *Self-organization of spiking neurons using action potential timing*. IEEE Transactions on Neural Networks, vol. 9, no. 3, pages 575–578, 1998.

[Schillen 91] T. B. Schillen & P. König. *Stimulus-dependent assembly formation of oscillatory responses : II. Desynchronization*. Neural Computation, no. 55, pages 155–166, 1991.

[Schneider 69] G.E. Schneider. *Two visual systems : brain mechanisms for localization and discrimination are dissociated by tectal and cortical lesion*. Science, vol. 163, pages 895–902, 1969.

[Searle 98] J. Searle. *Minds, Brains and Programs*. The behavioural and brain science, no. 3, pages 417–424, 1998.

[Senn 01] W. Senn, H. Markram & M. Tsodyks. *An Algorithm for Modifying Neurotransmitter Release Probability Based on Pre- and Postsynaptic Spike Timing*. Neural Computation, vol. 13, no. 1, pages 35–67, 2001.

[Shannon 50] C. Shannon. *A Chess-Playing Machine*. Scientific American, no. 182, pages 48–51, 1950.

[Singer 93] W. Singer. *Synchronization of cortical activity and its putative role in information processing and learning*. Annual Review of Physiology, no. 55, pages 349–374, 1993.

[Steels 95] L. Steels. *When are robots intelligent autonomous agents ?* Journal of Robotics and Autonomous Systems, vol. 15, pages 3–9, 1995.

[Steels 96] L. Steels. *Perceptually grounded Meaning Creation*. In M. Tokoro, editeur, Proceedings of the International Conference on Multiagent Systems (ICMAS-96), pages 338–344, Menlo Park, CA, 1996. AAAI Press.

[Steels 97a] L. Steels. *Construction and Sharing Perceptual Distinctions*. In M. van Someren & G. Widmer, editeurs, Proceedings of the European Conference on Machine Learning, Berlin, 1997. Springer Verlag.

[Steels 97b] L. Steels. *The synthetic modeling of language origins*. Evolution of Communication, vol. 1, no. 1, pages 1–34, 1997.

[Steels 98] L. Steels. *The origins of syntax in visually grounded robotic agents*. Artificial Intelligence, vol. 103, pages 1–24, 1998.

[Steels 01] L. Steels. *The Methodology of the Artificial*. Behavioral and Brain Sciences, vol. 24, no. 6, 2001. A reply to Webb, B. (2001) Can robots make good models of biological behavior ? *Behavioral and Brain Sciences*, 24(6).

[Steels 02] L. Steels, F. Kaplan, A. McIntyre & J. Van Looveren. *Crucial factors in the origins of word-meaning*. In A. Wray, editeur, The Transition to Language. Oxford University Press, Oxford, UK, 2002.

[Sutton 98a] R. S. Sutton & A. G. Barto. Reinforcement learning : An introduction. MIT Press, Cambridge, MA, 1998.

[Sutton 98b] R.S. Sutton & A.G. Barto. Reinforcement learning : An introduction. The MIT Press, Cambridge, MA, 1998.

[Thorndike 11] E. L. Thorndike. Animal intelligence. Hafner, Darien, CT, 1911.

[Thorpe 98] S. Thorpe & J. Gautrais. *Rank Order Coding : A new coding scheme for rapid processing in neural networks*. In Computational Neuroscience : Trends in Research, J. Bower, Ed, (Plenum Press, New York), pages 113–118., 1998.

[Thorpe 00] S. Thorpe, A. Delorme, R. VanRullen & W. Paquier. *Reverse Engineering of the Visual System Using Networks of Spiking Neurons*. In International Symposium on Circuits and Systems, pages 405–408, 2000.

[Thrun 98] Sebastian Thrun. *Learning Metric-Topological Maps for Indoor Mobile Robot Navigation*. Artificial Intelligence, vol. 99, no. 1, pages 21–71, 1998.

[Thrun 99] S. Thrun. *Learning maps for indoor mobile robot navigation*. Artificial Intelligence, 1 :21 to 71, 1999.

[Turing 70] A. Turing. *Intelligent Machinery*. Machine Intellligence, vol. Vol. 5, 1970.

[Ungerleider 82] L.G. Ungerleider & M. Mishkin. Two cortical visual systems. D. J. Ingle, M. A. Goodale and R. J. W. Mansfield (eds) Analysis of Visual Behavior, MIT Press, Cambridge, 1982.

[Ungerleider 94] L.G. Ungerleider & J.V. Haxby. *'What' and 'where' in the human brain*. Current Opinion Neurobiology, vol. 4, pages 157–165, 1994.

[Verschure 03] P. Verschure & P. Althaus. *A real-world rational agent : unifying old and new AI*. Cognitive Science, vol. 27(4), pages 561–590, 2003.

[Wermter 00] S. Wermter & R. Sun. Hybrid neural systems, volume 1778. Springer-Verlag Inc., New York, NY, USA, 2000.

[Wiener 43] N. Wiener. Cybernetics : or control and communication in the animal and the machine. MIT press, Cambridge, 1943.

[Wilson 89] M. A. Wilson, U. S. Bhalla, J. D. Uhley & J. M. Bower. *Genesis : A system for simulating neural networks*. Advances in Neural Information Processing Systems, vol. 1, pages 485–492, 1989.

[Yang 03] L. Yang & M. Jabri. *Sparse Visual Models for Biologically Inspired Sensory-Motor Control*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.

[Zhang 03] Y. Zhang & J. Weng. *Conjunctive Visual and Auditory Development via Real-Time Dialogue*. In C.G. Prince, editeur, Epigenetic Robotics, Modeling Cognitive Development in Robotic Systems, 2003.