Rapport de stage

Apprentissage neurorobotique pour la prédiction des affordances d'objets 3D et leur préhension et manipulation

Auteur :
M. Elias Aoun Durand

Jury :
Pr. L. Hardouin
Pr. L. Jaulin
Pr. B. Zerr
Pr. F. Lebars
Pr. S. Rohou

Version du
27 août 2019

Table des matières

Table des figures

Introduction

0.1 Présentation du domaine d'étude

En robotique, lier les actions possibles aux objets, ce qu'on appelle des affordances est encore une tâche difficile. Cela demande non seulement d'apprendre comment le corps se meut par rapport à lui-même mais également comment le corps se meut relativement à des objets en mouvement avec des orientations et des positions différentes.

C'est pour réaliser des tâches de saisies, pour représenter le corps dans l'espace et les objets à portée de main, ce qui correspond à l'espace péri-personnel.

Le cortex pariétal est responsable de l'encodage des relations spatiales des couples corps/objets en intégrant des informations multimodales. En effet, le cortex pariétal encode la distance et l'orientation relative qui permet de représenter les objets dans des coordonnées relatives centrées sur les parties du corps qui servent à les saisir ou les sentir.

Le mécanisme qui permet de fusionner les informations visuelle, proprioceptive, tactile, vestibulaire et motrice est le mécanisme de modulation en gain ou "gain field mechanism" qui implémente les transformations sensorimotrices depuis un repère global vers des coordonnées relatives à partir de représentations mixtes multimodales (exemple : coordonnées d'une cible centrés sur la main dans le repère du système visuel).

D'un point de vue computationnelle, ces représentations mixtes sont intéressantes car elles peuvent servir à générer des modèles de cinématiques inverses et directs dans divers repères de références dans l'espace à partir des celles entrées des senseurs.

0.2 Objectifs du stage

Mon stage a consisté à modéliser ces fonctions du cortex pariétal à travers trois architectures et expériences robotiques dans l'objectif d'apprendre des cellules neuronales motrices en 3D, l'encodage 3D de mouvement visuel, l'alignement entre la main et l'objet pour la saisie 3D avec l'intégration multimodale des informations proprioceptive, motrice et visuelle en utilisant le mécanisme de modulation de gain.

Dans une première expérience avec un robot modélisé, j'ai proposé un modèle de cellules direction-visuel-moteur trouvé par Georgopoulos [5] dans les années 1980 en utilisant le mécanisme de modulation de gain. Ces neurones moteurs sont sensibles aux directions visuelles et sont responsables de la saisie en 3D ainsi que de l'apprentissage de primitives motrices.

Dans une deuxième expérience en utilisant l'information visuel et motrice, j'ai développé un réseau de neurones qui exploite une représentation mixte pour apprendre les transformations visuelles affines relative à des mouvements 3D dans l'espace. Cette fonction, trouvée dans le cortex pariétal par Kakei [6], est particulièrement intéressante pour détecter les mouvements d'objets 3D et les mouvements de la main ; ce réseau est important pour le contrôle du mouvement ainsi que pour le choix des actions futures à prendre.

Dans une troisième expérience, j'ai développé une architecture qui combine les deux précédentes pour contrôler et aligner la main robotique à des orientations spatiales particulières correspondant à des localisations et orientations cibles.

0.3 Organisation du rapport

Dans un premier temps, en partant d'observations sur les neurones du cortex pariétal et d'éléments de machine learning, je vais définir un type de réseau de neurones qui va être utilisé pour apprendre les transformations sensori-motrices nécessaires à la préhension et manipulation d'un objet en 3D. Ensuite, je vais décrire les expériences que j'ai réalisé pour que le réseau apprennent les fonctions voulues. De plus, je vais m'attarder sur des réseaux que j'ai également développés et comparer leur fonctionnement. Dans cette seconde partie, j'analyserais également quelques résultats obtenus avant de conclure.

0.4 Organisme d'accueil

Le laboratoire ETIS (Equipes Traitement de l'information et Système) est une unité de recherche mixte commune au CNRS (UMR 8051), à l'ENSEA Cergy et à l'Université de Cergy Pontoise. Il est rattaché à l'institut des sciences informatiques et leurs interactions. L'équipe neurocybérnetique est l'équipe dans laquelle j'ai travaillée. Elle a pour but de faire de la robotique développementale et bio-inspirée. Le laboratoire est composé de quatre équipes de recherche :

1. Indexation Multimédia et Intégration de données (MIDI)
2. Information, Communications, Imagerie (ICI)
3. Architectures et Technologies pour les unités Reconfigurables Embarquées (ASTRE)
4. Neurocybernétique

Chapitre 1

"Gain-Field Networks"

1.1 Transformations sensorimotrices dans le cortex pariétal

1.1.1 Intégration multimodale dans le cortex pariétal

Pour se déplacer dans l'espace, il est nécessaire de corréler les variations se produisant dans le domaine sensoriel à celles se produisant dans le domaine moteur. Ces corrélations ou transformations sensorimotrices sont apprises tout au long de la vie pour estimer où se situe visuellement le corps et comment contrôler son mouvement relatif pour atteindre un point dans l'espace. Une fois apprises, ces transformations peuvent être utilisées pour anticiper le contact avec des objets, estimer la position du corps sans information visuelle ou adapter les commandes motrices en fonction du changement de schéma corporel (pour l'utilisation d'un outil par exemple), [1], [3].

Dans le cerveau, des circuits neuronaux du cortex pariétal sont impliqués dans l'apprentissage des corrélations entre des signaux multimodaux (tactile, visuel, proprioceptif, vestibulaire, auditif)[4, 9, 7]. L'observation de ces neurones moteurs multimodaux a permis de montrer que leur activité dépend non seulement d'une activité motrice préférentiel mais également d'une activité sensoriel préférentiel et l'amplitude de cette activité encode une distribution jointe de ces signaux [7]. Par exemple, certains neurones vont avoir une activité maximale pour un mouvement dans une certaine direction mais également pour une certaine position de l'oeil.

Comme ces neurones sont sensibles à différents types d'informations sensoriels on les nomment "gain-field neurons". En effet, tout se passe comme si l'activité du neurone moteur était modulé (gain) par l'information sensoriel.

Ces neurones vont permettre de combiner les informations sensorielles afin de construire

des systèmes de références attachés à certaines parties du corps (centrés sur la tête, les yeux, la main, etc.) Par exemple, certains neurones vont être sensibles à tout objet entrant dans le champ visuel et s'approchant de la main, quelque soit la position de l'oeil. Cela signifie qu'une conjonction de ces neurones multimodaux a permis la construction d'un repère centré sur la main.

1.1.2 Fonctions du cortex pariétal

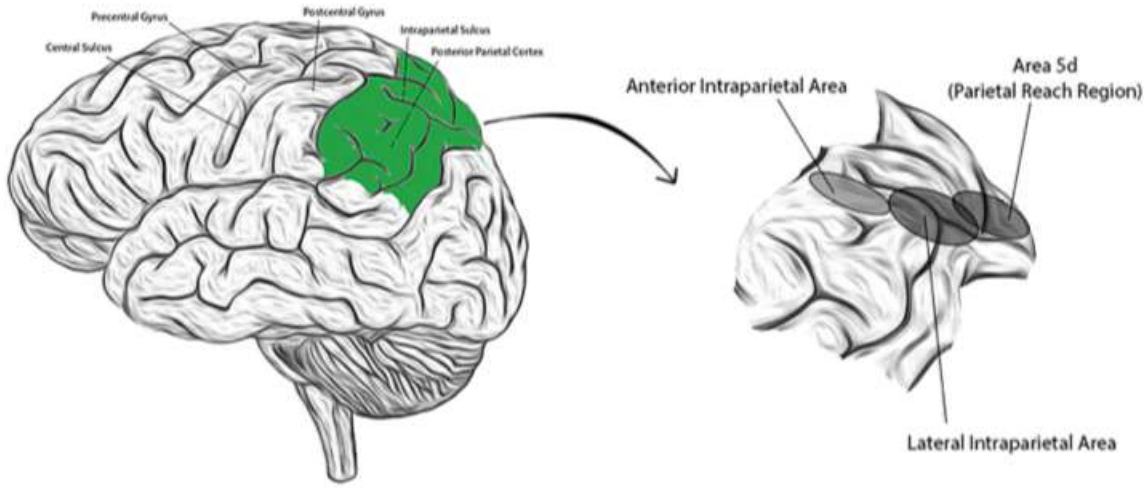

FIGURE 1.1 – Vue latérale du cortex pariétal humain.

Dans cette section on décrit plus en détail les régions fonctionnel et anatomiques du cortex pariétal [18]. Le cortex pariétal postérieur (PPC en anglais) comprend divers sous-régions qui reçoivent et intègrent des entrées multimodales (visuelle, auditives, somatosensorielles, vestibulaire) qui permet la représentation de fonctions cognitives abstraites. Des régions du PPC ont été identifiées avec des concentrations de neurones sensibles à la prise de décision, aux mouvements de saisie, à la préhension et à la représentation du corps et de l'environnement immédiat.

Aire intrapariétal latéral

Cette région encode des activités reliés à des mouvements oculaires. C'est un site de l'apprentissage de catégorisation de divers attributs visuels, et en particulier de direction du mouvement et de forme. Il est également le lieu d'activité liés à la prise de décision et de synchronisation de mouvements oculaires vers une cible.

Aire intrapariétal antérieur

Cette région est sensible aux objets et aux postures de la main pour les saisir [14]. Son inactivation empêche la coordination motrice des doigts durant la saisie d'un objet. Un petit nombre de ces neurones ont permis de décoder tout un ensemble de positions de la main pour la saisie, encodées dans un repère centré sur la main.

Aire 5d

Cette région encode la position d'une cible dans un repère centré sur la main et est liée aux trajectoires et aux mouvements vers les cibles. Il contient une forte concentration de neurones qui intègrent une information multimodale. Elle est également impliquée dans la représentation d'un schéma corporel et de l'espace péri-personnel. Il est également possiblement le siège de l'interprétation des gestes réalisés par autrui (grâce aux "neurones miroirs").

1.1.3 Fonctions à base radiale (RBF)

Dans cette section ont décrit les implications liées à la présence de transformations sensorimotrices dans le cerveau. Les transformations sensorimotrices sont des applications non linéaires réalisées dans le cortex pariétal [15]. Considérons par exemple le calcul de la position $\overrightarrow{A}$ d'un objet dans le repère centrée sur la tête en fonction de sa position sur la rétine $\overrightarrow{R}$ et de la position de l'oeil dans ce même repère $\overrightarrow{E}$ (figure 1.2). On a :

$$\overrightarrow{A} = \overrightarrow{R} + \overrightarrow{E} \tag{1.1}$$

est en dimension 3 une équation non linéaire puisque la rotation en dimension 3 est non linéaire. Or le cerveau n'a pas accès à $\overrightarrow{R}$ comme a une liste de nombre (composantes verticale et horizontale). En effet le champ réceptif des neurones du cortex prémoteur et du système visuel sont typiquement des gaussiennes. Ces champs réceptifs sont en fait des fonctions non linéaires qui dépendent de $\overrightarrow{R}$. Puget a proposé que les neurones du cortex pariétal encode des fonctions à base radiale non linéaires des entrées. Les fonctions à base radial sont un type de fonction qui va permettre d'approcher des fonctions non linéaires.

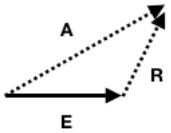

FIGURE 1.2 – A est centré sur le centre de la tête, R est centré sur le centre de la rétine (les pointillés signifient la profondeur).

Un exemple de fonction à base radiale

Cette exemple est tiré du travail de Pouget [15]. Considérons la fonction exponentielle e^x. Une manière de l'approcher peut être de calculer son développement en série de Taylor. Mais ce n'est pas la seule manière d'approcher cette fonction. En effet, on peut l'exprimer comme une combinaison linéaire de sinus et de cosinus pondérée par des poids (série de Fourier). Si on a à disposition une bonne approximation des fonctions cosinus et sinus, il n'est plus nécessaire de les recalculer à chaque fois, et le résultat est une combinaison linéaire de ces fonctions. De plus, les séries de Fourier peuvent approcher un grand nombre de fonctions non linéaires.

On appelle dans ce contexte fonctions à base radiale ces famille de fonctions (comme (cos, sin)) dont la somme pondérée permet d'approcher des fonctions non linéaires. L'hypothèse de Pouget est que les neurones du cortex pariétal implémentent une base appropriée aux transformations sensorimotrices. Une commande motrice devient alors *une combinaison linéaire* des différents champs réceptifs. Si une commande motrice M est une fonction non linéaire de entrées sensorielles, par exemple visuelle V et proprioceptive P (c'est à dire de position du corps) alors on a :

$$M = \sum_i c_i B_i(V, P) \tag{1.2}$$

où les c_i sont des coefficients qui dépendent de la commande M à calculer (coefficients de Fourier si les fonctions de base B_i sont des sinus et cosinus). Les réponses des neurones pariétaux se comportent comme les fonctions de base $B_i(V, P)$.

Il y a plusieurs avantages à une telle décomposition. Une fois que ces fonctions ont été calculées, le nombre d'opérations nécessaires pour obtenir une commande motrice est fortement réduit puisque n'importe quelle application non linéaire des entrées ne nécessite plus qu'une combinaison linéaire de ces fonctions, c'est à dire une simple projection. Ensuite l'activité de ces neurones peut être utilisée pour calculer plusieurs commandes motrices différentes. Enfin, il n'est nécessaire d'apprendre ces fonctions à base radiale qu'une seule fois et de manière non supervisée.

Pouget propose d'utiliser comme fonctions à base radiale la fonction gaussienne et sigmoïde.

1.2 Fonctionnement et modélisation des "gain-field neurons"

1.2.1 Modélisation d'un neurone

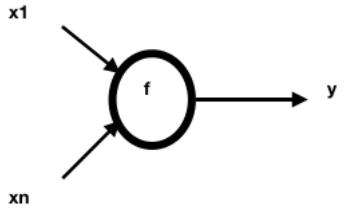

FIGURE 1.3 – Modèle simplifié d'un neurone

Soient $(x_1, ..., x_n) \in \mathbb{R}$ les entrées pré-synaptiques d'un neurone et $(w_1, ..., w_n)$ les poids de connexion associés. L'activité y réalise une fonction non linéaire et bornée :

$$y = f(x_1, ..., x_n, w_1, ..., w_n) \tag{1.3}$$

Les neurones modélisés en machine learning peuvent par exemple avoir pour activité :

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + b) \tag{1.4}$$

où σ est une fonction non linéaire, $\sigma(x) = max(0, x)$ pour fixer les idées, et b un biais.

1.2.2 Fonctionnement

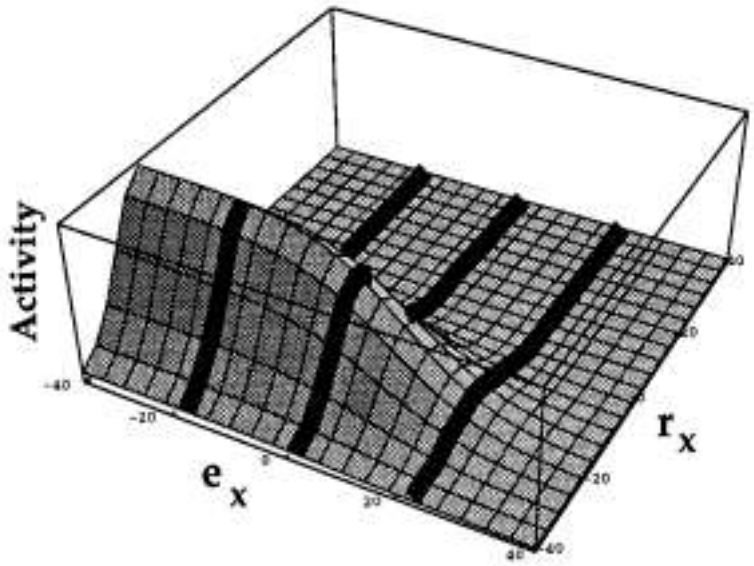

FIGURE 1.4 – Exemple d'activité neuronale bi-modale.

La particularité des neurones multimodaux décrit ci-dessus est que **l'activité en sortie dépend multiplicativement de ses entrées**. Par exemple, supposons qu'un neurone est deux entrées pré-synaptiques (x_1, x_2) alors son activité en sortie y va s'exprimer comme le produit des deux entrées pondéré par un poids w :

$$y = \sigma(wx_1x_2) \tag{1.5}$$

On peut distinguer deux situations. La première où x_1 agit comme un interrupteur en prenant soit 0 soit 1 comme valeur, et ce système est analogue à un transistor. La seconde, plus générale et exprimer ci-dessus est la modulation (gate) par x_1 du signal entre x_2 et y . La figure 1.4 montre un exemple d'amplitude de l'activité d'un neurone du cortex pariétal qui dépend à la fois de la position de l'oeil e_x et de la position dans le repère de l'oeil d'une cible à saisir r_x : on remarque bien la forme de la gaussienne modulée par une sigmoïde.

1.2.3 Modélisation des "gain-field neurons" et généralisation

Les informations qu'ont souhaitent corrélées (motrices, visuelles, etc.) ne dépendent pas, en général, d'une variable, c'est à dire qu'on va plutôt manipuler des vecteurs représentant une information visuelle $\mathbf{x} = (x_1, ..., x_{n_x})^T$ et une information motrice $\mathbf{z} = (z_1, ..., z_{n_z})^T$, dont les tailles sont a priori différentes. Ainsi, on a plus une activité mais un vecteur

d'activité $\mathbf{y} = (y_1, ..., y_{n_y})$ qu'on peut exprimer comme suit :

$$\forall j, y_j = \sigma_y(\sum_{i=1}^{n_x}\sum_{k=1}^{n_z} W_{ijk}x_i z_k) \tag{1.6}$$

Les poids W_{ijk} définissent un tenseur d'ordre 3. Si n_x, n_y et n_z sont du même ordre n, alors le nombre de poids est cubique en n. Computationnellement ce n'est pas satisfaisant et on verra par la suite comment y remédier.

1.3 "Gated networks"

En fait, le mécanisme définit ci-dessus n'est pas spécifique à la modélisation neuronale et a été utilisé en machine learning pour relier deux entrées entre elles [10], [11]. On les nomment "gated networks" ou "gain-field networks" en anglais. Ils ont en effet certains avantages :

> Gated networks are extensions of deep learning building blocks that are designed to learn relationships between at least two sources of input and at least one output. When the relationships between several source of data involves multiplicative interaction, **such gating connections between neurons result in more natural topologies and increase the expressive power of neural networks**, because implementing a multiplicative relationship between two layers of standard neurons would require a number of dedicated neurons that would grow exponentially with the required precision.

Gated Networks : an inventory, *Sigaud, Masson, Filliat*

Ainsi pour mieux comprendre le fonctionnement des réseaux neuronaux du cortex pariétal qui fusionnent multiplicativement les informations multimodales, on va s'intéresser aux "gated networks" développer en apprentissage automatique.

1.3.1 Architecture standard

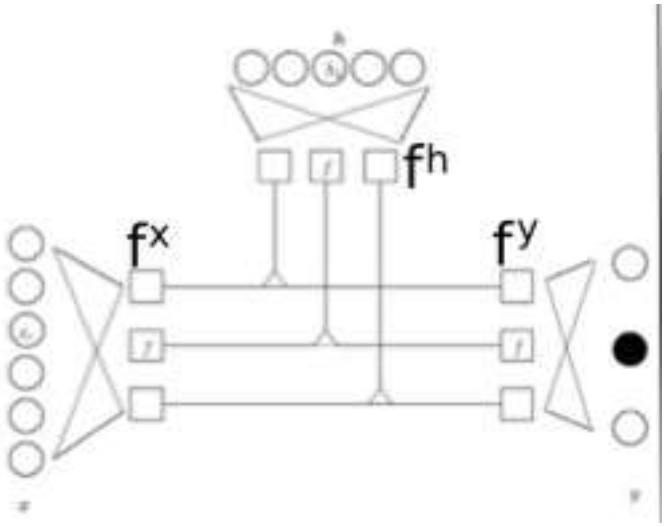

FIGURE 1.5 – Schéma d'un réseau à modulation de gain avec couches de factorisation

Originellement, les "gated networks" ont été utilisé par Memisevic [10, 11, 12, 13] pour apprendre à encoder des transformations visuelles. En partant de deux photos (par exemple, deux photos d'une même scène prise à quelques secondes d'intervalles, ou deux photos de visages pris sous deux angles différents), on cherche la fonction qui transforme la première en la deuxième. Implicitement, on espère que la fonction encodée comporte les informations nécessaires à la compréhension de la transformation : dans le cas de deux visages pris sous deux angles différents, on espère que la transformation encodée va contenir des informations sur la rotation et la translation qui ont amenées le changement de prise de vue final. (on verra par la suite qu'un tel réseau est en fait particulièrement adapté à l'apprentissage de transformations dans l'espace). De manière générale, ces réseaux vont permettre d'inférer par apprentissage la transformation y qui amène la donnée d'entrée x_1 à la donnée x_2. On parle également de représentation latente plutôt que de transformation, en effet, dans le cas des deux images prises à deux instants successifs, savoir comment passer d'une image à la suivante suppose qu'une certaine "condensation" des informations nécessaires à cette transformation a été réalisée dans le réseau.

Il est intéressant de remarquer qu'étant donné ce tenseur de poids, il est possible de reconstruire une estimation (symbolisée par l'accent circonflexe) d'une donnée connaissant les deux autres. C'est à dire qu'on a à la fois :

$$\forall j, \hat{y}_j = \sigma_y(\sum_{i=1}^{n_x}\sum_{k=1}^{n_z} W_{ijk}x_i z_k) \tag{1.7}$$

$$\forall i, \hat{x}_i = \sigma_x(\sum_{j=1}^{n_y}\sum_{k=1}^{n_z} W_{ijk} y_j z_k) \tag{1.8}$$

$$\forall k, \hat{z}_k = \sigma_z(\sum_{i=1}^{n_x}\sum_{j=1}^{n_y} W_{ijk} x_i y_j) \tag{1.9}$$

Sur la figure 1.5, on voit les deux entrées x et h et la sortie y ainsi que les couches de factorisation f_x, f_h et f_y. Une autre manière de représenter ce type de réseau est présenté sur la Figure 1.6 .

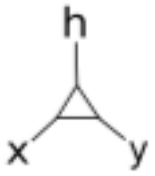

FIGURE 1.6 – Notation simplifiée correspondant au réseau décrit à la figure 1.5

1.3.2 Apprentissage

Ces réseaux ont deux entrées et une sortie. Une manière de les entraîner est d'utiliser l'apprentissage supervisé : pour une pair de données en entrées, on fournit la sortie attendue, et on entraîne le réseau à minimiser une fonction d'erreur calculée entre la sortie attendue et la sortie obtenue. Cette méthode suppose donc qu'on connaisse a priori les sorties pour entraîner le réseau. Dans notre cas, on préfère utiliser une méthode non supervisée.

En effet, on utilise plutôt une méthode non supervisée qui consiste, étant donné les deux données d'entrées, à reconstruire l'une des deux à l'aide d'un auto-encoder. On obtient alors un "gated auto-encoder", ou GAE [8], qui est au coeur de la modélisation réalisée lors de mon stage. Dans la pratique, on pourrait imaginer un robot qui apprend continuellement, c'est à dire qui n'a pas accès à une "vérité terrain" extérieur mais seulement à la donnée de ses capteurs et qui apprend à les reconstruire ou à prédire leurs valeurs futures.

1.3.3 Auto-encodeur

Un auto-encodeur se compose de deux fonctions :

1. Une fonction d'encodage qui transforme une donnée d'entrée $\mathbf{x}$ en une représentation latente $\mathbf{h}$, typiquement $\mathbf{h} = \mathbf{h}(\mathbf{x}) = \sigma_h(\mathbf{W}\mathbf{x} + b)$.
2. Une fonction de décodage qui reconstruit une représentation $\hat{\mathbf{x}}$ de $\mathbf{x}$ à partir de la représentation latente $\mathbf{h}$, typiquement $\hat{\mathbf{x}} = \mathbf{r}(\mathbf{h}) = \sigma_x(\mathbf{W}'\mathbf{h} + b')$.

On définit ensuite une fonction de coût qui dépend de l'erreur de reconstruction, par exemple $e = \| \hat{\mathbf{x}} - \mathbf{x} \|^2$. L'apprentissage consiste alors à appliquer un algorithme d'optimisation (descente de gradient) aux poids synaptiques du réseau pour minimiser la fonction de coût. Ainsi, pendant l'apprentissage, le réseau apprend simultanément la fonction d'encodage et de décodage en utilisant :

$$\hat{\mathbf{x}} = \sigma_x(\mathbf{W}'\sigma_h(\mathbf{W}\mathbf{x} + b) + b') \tag{1.10}$$

Et on va donc chercher $\mathbf{W}$ et $\mathbf{W}'$ qui minimise e. L'intérêt d'utiliser un autoencodeur est également qu'il apprend à la fois un modèle direct (via l'encodeur : depuis l'information sensorielle vers une représentation latente) et un modèle invers (via le décodeur : depuis la représentation latente vers une reconstruction de l'information sensorielle).

1.3.4 Factorisation du tenseur

Comme on l'a vu précédemment, le tenseur de poids $(W_{ijk})_{1\leq i\leq n_x, 1\leq j\leq n_y, 1\leq k\leq n_z}$, qu'il est nécessaire d'optimiser, peut être de très grande dimension : imaginons que chacune des entrées soit une image avec 10^6 pixels, la taille du tenseur est alors au moins de 10^{12} c'est à dire plus que le nombre d'atomes dans l'univers ! On décrit dans cette section une méthode pour réduire la taille de ce tenseur.

On cherche à factoriser le tenseur de poids, afin de réduire le nombre de paramètres à optimiser. On peut y parvenir de deux manières :

1. Par projection des entrées et sorties sur un espace plus petit à l'aide de couches de factorisation dans le réseau (factor layors).
2. En imposant des contraintes sur le tenseur global afin de réduire directement le nombre de poids à optimiser.

Projection

Cette approche consiste à "remplacer" les entrées $\mathbf{x} = (x_1, ..., x_{n_x})^T$ (et également $\mathbf{y}$ et $\mathbf{z}$) par un vecteur plus petit $\mathbf{f}^x = (f^x_1, ..., f^x_{n_f})^T$ (resp. $\mathbf{f}^y$, $\mathbf{f}^z$) avec $n_{f^x} < n_x$ (resp. $n_{f^y} < n_y$, $n_{f^z} < n_z$).

Cette opération est réalisée en faisant passer les entrées et sortie à travers des réseaux de neurones denses (dont le nombre de paramètres à optimiser est moindre que pour le tenseur $\mathbf{W}$).

On peut également utiliser d'autres types de réseaux de neurones pour cette projection. Par exemple, si les données d'entrées sont des images, il sera plus judicieux d'employer des réseaux de neurones convolutifs. En effet, cette opération de projection consiste en réalité à réduire la dimension de l'espace des données d'entrées et d'extraire des "features" dans un espace intermédiaire de dimension moindre. Or on sait que les réseaux de neurones convolutifs sont mieux adaptés aux problèmes d'extraction de features pour des images.

On verra dans les expériences menées que j'ai utilisé à la fois des réseaux convolutifs quand les entrées ou les sorties sont des images et des réseaux denses quand les entrées ou les sorties sont d'une autre nature.

Le nombre de paramètres du tenseur à optimiser devient donc $n_{f_x} * n_{f_y} * n_{f_z}$ auquel se rajoute les paramètres des réseaux extracteurs de features respectivement $n_x * n_{f_x}, n_y * n_{f_y}, n_z * n_{f_z}$, ce qui donne un nombre de paramètres totale de $n_{f_x} * n_{f_y} * n_{f_z} + n_x * n_{f_x} + n_y * n_{f_y} + n_z * n_{f_z}$

En résumé, les données $\mathbf{x}, \mathbf{z}$ et $\mathbf{y}$ sont d'abord projetées dans un espace réduit à travers des couches de factorisation, puis on réalise le produit tensoriel et finalement on reprojette la sortie du produit dans une dernière couche de "défactorisation".

Il est intéressant de remarquer qu'on peut encore réduire le nombre de poids en partageant les poids des couches de factorisation si les données sont de même nature. Par exemple, dans le cas d'images, on peut partager les poids entre les couches de factorisation de $\mathbf{x}$ et $\mathbf{z}$.

Contrainte

L'autre manière de réduire le nombre de paramètres est de contraindre la forme que prend le tenseur. Plus spécifiquement, on peut avoir un tenseur de la forme :

$$W_{ijk} = \sum_{f=1}^{F} W_{if}^{x} W_{jf}^{y} W_{kf}^{z} \tag{1.11}$$

En posant :

$$\begin{aligned} f_f^x &= \sum_{i=1}^{n_x} W_{if}^x x_i, \\ f_f^y &= \sum_{i=1}^{n_y} W_{if}^y x_i \\ f_f^z &= \sum_{i=1}^{n_z} W_{if}^z x_i \end{aligned} \tag{1.12}$$

On obtient :

$$\forall j, \hat{y}_j = \sigma_y(\sum_{i=1}^{n_x}\sum_{k=1}^{n_z} W_{if}^y f_f^x f_f^z) \tag{1.13}$$

Plus simplement on peut écrire que :

$$\begin{aligned} \mathbf{f}^x &= \mathbf{W}^{x\mathbf{T}}\mathbf{x} \\ \mathbf{f}^y &= \mathbf{W}^{y\mathbf{T}}\mathbf{y} \\ \mathbf{f}^z &= \mathbf{W}^{z\mathbf{T}}\mathbf{z} \end{aligned} \tag{1.14}$$

Et l'équation (1.11) devient :

$$\hat{\mathbf{y}} = \sigma_y(\mathbf{W}^y(\mathbf{f}^x \otimes \mathbf{f}^z)) \tag{1.15}$$

où $\otimes$ désigne une multiplication terme à terme.

1.3.5 Représentation latente

Il est intéressant de remarquer que les étapes de factorisation présentes des similarités avec le comportement en fonction à base radiale des neurones du cortex pariétal. En effet, la dernière couche présente les caractéristiques souhaitées :

1. la réponse de chaque neurone est une fonction non linéaire des entrées.
2. une fois l'apprentissage terminé, ces fonctions ne sont pas modifiées.
3. les commandes motrices obtenues en sortie du réseau sont une combinaison linéaire des réponses de chaque neurone.

.

Ainsi pour des nécessités de facilité de calcul on a introduit des couches de factorisation qui forment une base à partir de laquelle, quand l'apprentissage est terminé, les commandes motrices sont calculées par combinaison linéaire. On appellera primitives visuo-motrices les sorties de la dernière couche de factorisation, le terme primitive faisant référence au fait que chaque neurone n'actionne pas un moteur en particulier mais que c'est bien une combinaison de ceux-ci qui permet d'avoir une représentation du mouvement et donc des commandes à effectuer.

1.3.6 Performance

On sait que minimiser le carré de l'erreur de reconstruction est un bon critère d'entraînement qui favorise l'encodage, dans la représentation latente, de sous-ensembles dans lesquels résident la plupart des données . C'est à dire que ce critère est suffisant [12], [13] pour assurer que l'apprentissage du réseau va être une extraction pertinente de l'information contenue dans les données.

1.3.7 Résumé

Pour résumé, en partant de l'observation du caractère multiplicatif de l'amplitude de l'activité des neurones du cortex pariétal en fonction de ces entrées pré-synaptiques, on construit un modèle de réseau de neurones qui reproduit cette multiplcativité, qui se trouve être connu dans la littérature du machine learning comme des "gated networks". Partant de la nécessité d'un apprentissage non-supervisé dans notre application, on étoffe le réseau en proposant un "gated auto-encoders" qui s'optimise en reconstruisant ces entrées. En définissant la fonction d'estimation, on s'aperçoit qu'intervient un tenseur d'ordre 3 qu'il va être difficile de manipuler au vu des dimensions des données considérées et on décide donc de factoriser ce tenseur. En le factorisant, on introduit des couches supplémentaires soit convolutives soit denses. Finalement, on obtient un modèle assez général (au sens où les données d'entrée peuvent être des images, des coordonnées et des vecteurs de taile quelconque) qui va permettre d'inférer des corrélations entre des jeux de données et plus spécifiquement d'inférer des transformations qui amènent une donnée vers une autre. Il reste à définir les entrées et sortie pour les expériences qui suivent, ce qui est l'objet du chapitre suivant.

1.4 Affordance

Une affordance désigne un concept développé par Gibson [20] et qui associe une action, un acteur, un effet et un objet [17]. On peut voir une affordance comme une potentialité de réaliser telle ou telle action sur un objet, voir la figure 1.7. Un robot capable d'apprendre les affordances liées à son environnement est capable d'avoir un comportement autonome [16], [19].

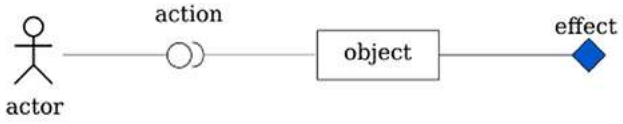

FIGURE 1.7 – Représentation d'une affordance.

Dans notre cas l'acteur est le bras robotique, l'action à réaliser est une action de saisie et/ou de manipulation, l'objet n'est pas déterminé, il s'agit dans un premier temps de voir comment on peut se diriger et s'orienter vers un point-cible, l'effet est observé à travers la caméra et les capteurs de positions.

Chapitre 2

Expérimentations

2.1 Setup

Les expériences vont faire intervenir :

1. un bras robotique à n degrés de libertés et dont la position de l'effecteur est notée (x_e, y_e, z_e).

2. une caméra fournissant des images à chaque instant t, $I(t) \in \mathbb{R}^{m*m}$ où m désigne le nombre de pixels.

3. des capteurs de positions fournissant les angles (θ_i) des n articulations.

4. les actionneurs du bras dont on connaît les commandes (m_i) (c'est à dire des vitesses angulaires).

Dans un premier temps, j'ai utilisé un modèle de bras robotique en "fil de fer" à trois degrés de liberté en Python qu'on peut voir sur la figure 2.1. Pour chacune des expérimentations, j'ai réalisé une base de données. J'ai effectué l'apprentissage, j'ai évalué cet apprentissage et analysé les résultats obtenus.

On a en fait trois types d'informations, des informations :

1. proprioceptives, notées P

2. motrices, notées M

3. visuelles, notées V

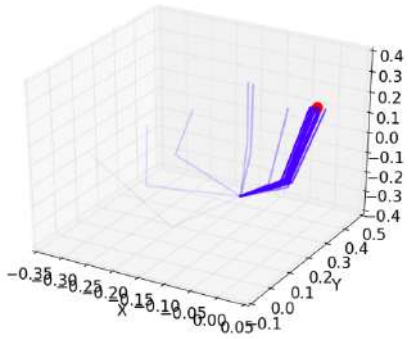

FIGURE 2.1 – Le bras dit "fil de fer".

2.2 Reconstruction ou prédiction ?

Considérons un triplet d'information (P, V, M), c'est à dire, pour fixer les idées, la donnée d'un vecteur d'angles des actionneurs, d'une image (tableau de pixels) et d'un vecteur de commandes motrices. Comme on l'a vu précédemment, on peut reconstruire indifféremment M connaissant P et V mais également reconstruire P connaissant M et V, etc., on parle alors de **reconstruction** (on continuera par la suite, à noter (P, V, M) le "gated network" qui a pour entrées (P, V) et pour sortie M).

Or les données dont nous parlons dépendent du temps ; c'est à dire que l'on a plus précisément $(P(t), V(t), M(t))$.

On peut non seulement utiliser deux informations hétérogènes (visuelle et proprioceptive) pour reconstruire la commande motrice, mais on peut également utiliser deux informations homogènes visuelles, $V(t)$ et $V(t+\Delta t)$, ou proprioceptives décalées dans le temps, pour reconstruire la commande motrice $M(t)$ qui a déplacée le bras entre l'instant t et l'instant $t+\Delta t$ et qui a amené le changement de perception visuel entre $V(t)$ et $V(t+\Delta t)$, on parle alors de **prédiction**, que l'on va noter $(V(t), V(t+\Delta t), M(t))$.

Toutes les combinaisons possibles ne sont pas forcément pertinentes, par exemple le triplet $(M(t), M(t+\Delta t), V(t+\Delta t))$: connaissant deux commandes motrices successives, il paraît illusoire de pouvoir reconstruire l'observation de la caméra après ces deux commandes. En effet, pour pouvoir prédire l'image obtenue par la caméra, il faut qu'il existe un lien entre la position du bras dans l'image et les commandes motrices successives. Or, a priori, une commande motrice peut être réalisée n'importe où dans l'espace de travail du bras de telle sorte que sa seule connaissance ne suffise pas à savoir où elle a été réalisée. Par contre, si une commande motrice (ou une série de commandes motrices successives) se trouvent être souvent réalisée au même endroit de l'espace de travail, alors il existe un lien entre les commandes et l'information visuelle, de telle sorte que le réseau peut ap-

prendre ce lien. On voit l'importance qu'il va falloir accorder à la répartition des données d'apprentissage dans l'espace de travail du bras robotique. On peut noter que le triplet $(M(t), V(t+\Delta t), M(t+\Delta t))$ est plus naturel.

Ci-dessous on trouve une liste des combinaisons utilisées dans les modèles présentés ci-après :

1. $(P(t), P(t+1), M(t))$; prédiction de la commande motrice en utilisant les capteurs d'angles.
2. $(V(t), V(t+1), M(t))$; prédiction de la commande motrice en utilisant la caméra.
3. $(V(t), P(t+1), M(t))$; prédiction de la commande motrice en utilisant un mixte entre la caméra et les capteurs.
4. $(V(t), M(t), V(t+1)$; reconstruction de l'image après la commande motrice effectuée.
5. $(P(t), M(t), V(t+1))$; reconstruction de l'image en utilisant seulement la proprioception.
6. $(V(t), M(t), P(t+1))$;reconstruction de la proprioception en utilisant seulement la caméra.

2.3 Implémentation : Tensorflow

L'implémentation des réseaux de neurones s'est faite en utilisant le framework Tensorflow qui offre des facilités pour créer des modèles de réseaux de neurones. La figure 2.2 montre une implémentation d'un encodeur pour un GAE où chaque ligne est une couche du réseau.

```
def build_dense_encoder(custom_shape=INPUT_ENCODER_SHAPE):

    inputs = tf.keras.Input(shape=custom_shape, name='encoder_input')

    x = tf.keras.layers.Lambda(lambda x: x[:, :, :, 0])(inputs)
    x = tf.keras.layers.Reshape((IMG_SIZE, IMG_SIZE, 1))(x)

    y = tf.keras.layers.Lambda(lambda x: x[:, :, :, 1])(inputs)
    y = tf.keras.layers.Reshape((IMG_SIZE, IMG_SIZE, 1))(y)

    fx = tf.keras.layers.Flatten()(x)
    fx = tf.keras.layers.Dense(
        LATENT_DIM, activation='relu', name='latent_enc_fx1')(fx)
    #fx = tf.keras.layers.Dense(LATENT_DIM, activation = 'relu', name = 'latent_enc_fx2')(fx)
    #fx = tf.keras.layers.Dense(LATENT_DIM, activation = 'relu', name = 'latent_enc_fx3')(fx)
    fx = tf.keras.layers.Reshape((LATENT_DIM, 1,))(fx)

    fy = tf.keras.layers.Flatten()(y)
    fy = tf.keras.layers.Dense(
        LATENT_DIM, activation='relu', name='latent_enc_fy1')(fy)
    #fy = tf.keras.Dense(LATENT_DIM, activation = 'relu', name = 'latent_enc_fy2')(fy)
    #fy = tf.keras.Dense(LATENT_DIM, activation = 'relu', name = 'latent_enc_fy3')(fy)
    fy = tf.keras.layers.Reshape((1, LATENT_DIM,))(fy)

    matmul = tf.keras.layers.Multiply()([fx, fy])

    fh = tf.keras.layers.Flatten()(matmul)
    fh = tf.keras.layers.Dense(LATENT_DIM, name='latent_fh1')(fh)
    #fh = tf.keras.layers.Dense(LATENT_DIM, name = 'latent_fh2')(fh)
    #fh = tf.keras.layers.Dense(LATENT_DIM, name = 'latent_fh3')(fh)

    fx = tf.keras.layers.Reshape((1, LATENT_DIM,))(fx)
    fh = tf.keras.layers.Reshape((1, LATENT_DIM,))(fh)

    outputs = tf.keras.layers.Concatenate()([fx, fh])
    encoder = tf.keras.Model(
        inputs=inputs, outputs=outputs, name='encoder_model')

    return encoder
```

FIGURE 2.2 – Implémentation de l'encodeur d'un GAE.

2.4 Base de données

Comme on l'a vu dans le setup, le choix des données d'entraînement va être très important car il va conditionner les liens qui vont ensuite exister entre les différentes informations sensorielles et donc dans l'application qui va lier les entrées sensorielles au sorties motrices. Dans la première expérience, on va faire apprendre au réseau des primitives visuo-motrices dans la couche de factorisation. En fonction du type de mouvement dans la base de données, on peut se demander quelles primitives visuo-motrices va être apprises et comment les données en entrées influencent celles-ci. L'hypothèse de Pouget est que le type de fonctions à base radiale apprises par les neurones prémoteur du cortex pariétal sont indépendantes, après un apprentissage suffisamment long, des sorties. Dans notre cas, on peut se demander quel type de primitives vont être apprises si les mouvements dans la base de données sont aléatoires ou s'ils représentent des mouvements de préhension et de saisie. J'ai choisi une base de données composée de 5000 mouvements obtenus en appliquant à 50 positions de base, 100 commandes motrices à chacun des actionneurs. On obtient un ensemble de mouvements représenté par les images et les positions successives du bras. Les images subissent un pré-traitement qui consiste à ajouter du bruit gaussien sur toute l'image.

2.5 Premier réseau : couplage visuo-moteur

Cette première expérience vise à modéliser des cellules direction-visuel-moteur trouvé par Georgopoulos dans les années 1980 en utilisant le mécanisme de modulation de gain. Ces neurones moteurs sont sensibles aux directions visuelles et sont responsables de la saisie en 3D ainsi que de l'apprentissage de primitives motrices. La direction visuelle d'un mouvement correspond à la direction prise par l'effecteur. Pour faire cet apprentissage, on fournit au réseau des pairs d'images correspondant à différent déplacements visuels.

La Figure 2.3 montre le modèle de réseau utilisé. Les données d'entrée sont deux images du bras en fil de fer. La première étape consiste à passer chacune de ces images à travers un réseau profonds convolutifs (ConvNet), qui sont une succession de convolution suivi de max-pooling, je n'ai pas détaillé ici le choix des hyperparamètres, mais il s'agit d'une extraction de features. L'opération de max-pooling est une opération non linéaire de sous-échantillonnage. Pour ce premier réseau, la base de données a été formée à partir de 50 positions de bras répartie aléatoirement dans l'espace de travail, auquel j'ai appliqué 100 commandes différentes pour chaque position, ce qui fait un total de 5000 mouvements différents, et donc 5000 pairs d'images.

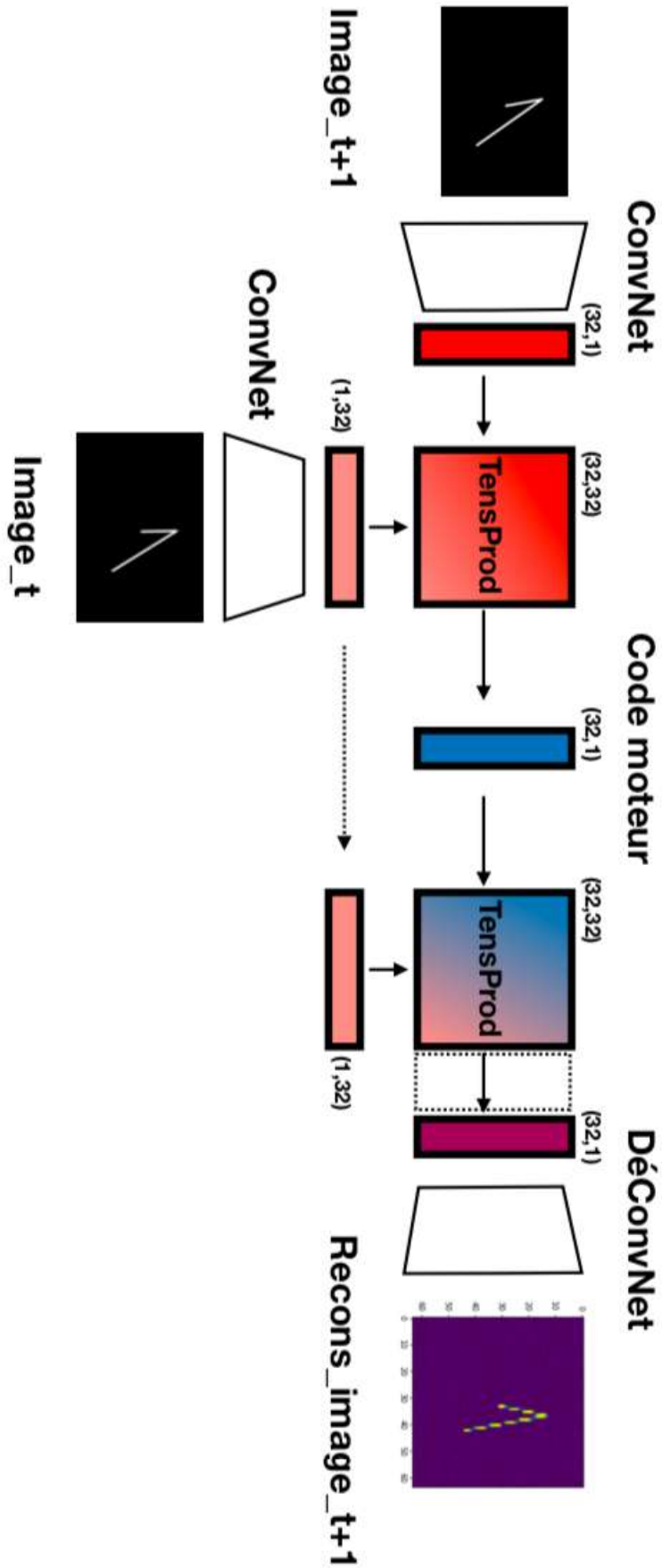

FIGURE 2.3 – Première expérience

Courbes d'apprentissage

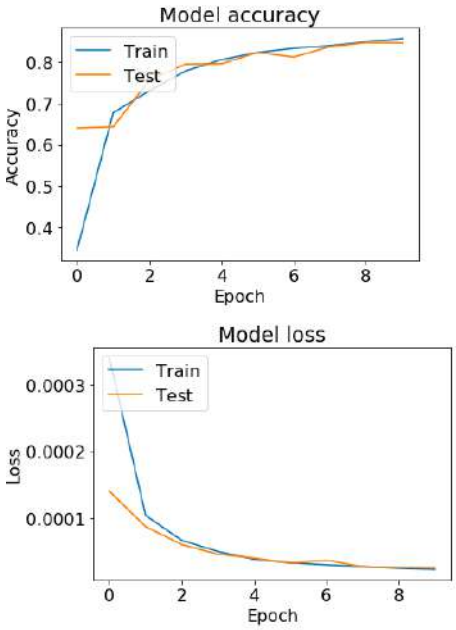

FIGURE 2.4 – Performance de l'apprentissage en fonction du nombre d'époques.

L'apprentissage est réalisé par lots (batchs) de 500 sur 10 époques.

Exemples d'images reconstruites

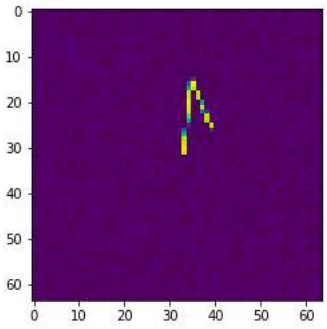

FIGURE 2.5 – bras fil de fer

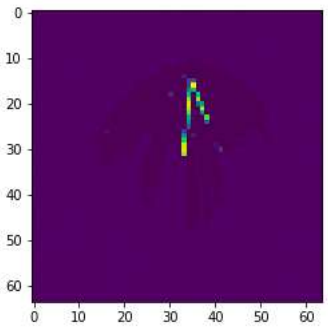

FIGURE 2.6 – bras fil de fer reconstruit

2.6 Deuxième réseau : commande visuo-motrice

Pour la deuxième expérience, on cherche à apprendre au réseau des transformations visuelles relatives à des mouvements 3D en utilisant l'espace des représentations latentes développé dans le réseau précédent. Pour chaque mouvement dans l'espace (pair d'images $(I(t), I(t+1))$) on va associer le vecteur obtenu après le passage dans le premier réseau qui correspond à une représentation mixte visuo-motrice liée à la direction visuelle du mouvement. La base de données est donc formée des 5000 représentations mixtes obtenues à l'étape précédente associé à la position $P(t)$ du bras avant que la commande soit effectuée. On apprend au réseau à reconstruire cette représentation mixte.

Courbes d'apprentissage

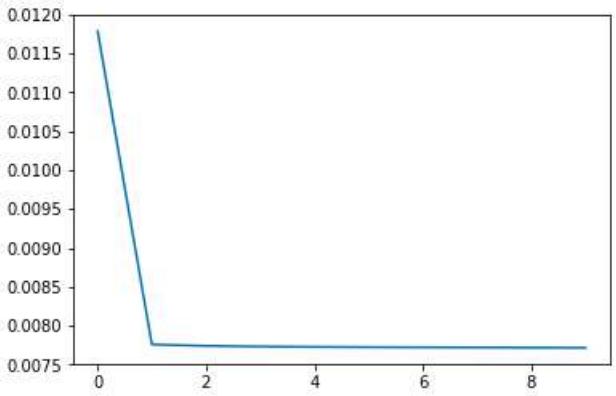

FIGURE 2.7 – Valeur de la fonction de coût en fonction du nombre d'époques.

Sur la figure 2.7 on voit que l'apprentissage converge au bout de 2 époques, ce qui peut laisser penser que la représentation latente est adaptée et qu'il est donc possible d'utiliser la représentation latente pour le contrôle puisqu'elle permet de discriminer efficacement les mouvements visuels à partir de leur représentation latente.

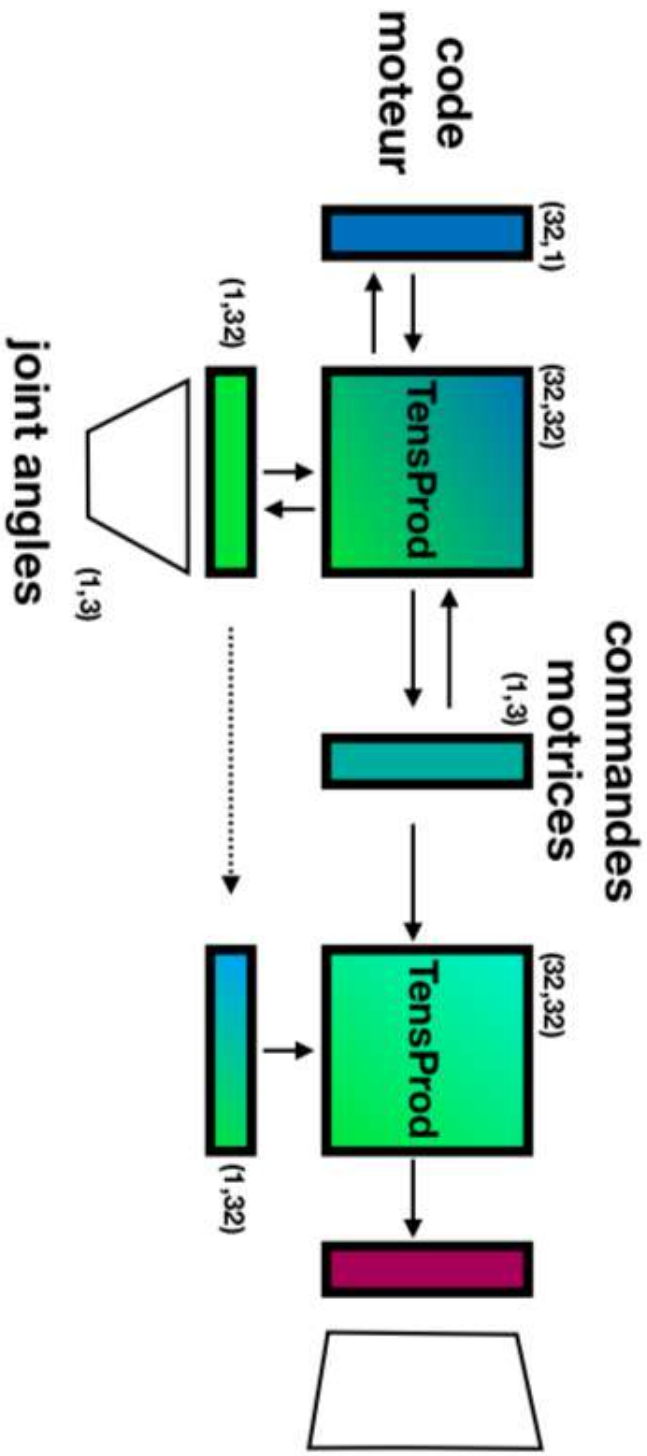

FIGURE 2.8 – Deuxième expérience

2.7 Troisième réseau : combinaisons des précédents réseaux

Le troisième réseau associe les deux premiers. On souhaite ainsi réaliser le contrôle du bras robotique en fournissant en entrée la direction visuelle dans laquelle on souhaite aller, et on espère qu'après l'apprentissage, la sortie du réseau nous donne une bonne estimation des commandes à appliquer aux actionneurs du bras pour lui permettre d'aller dans cette direction. Supposons qu'on souhaite déplacer le bras dans une certaine direction visuelle, on va trouver grâce au premier réseau, la représentation mixte qui correspond à ce mouvement particulier. On utilise cette représentation pour calculer une commande avec le second réseau. On applique ensuite la commande au système. On capte la nouvelle position du bras, ce qui nous donne une mesure de l'erreur commise à l'étape précédente. Puis on itère en utilisant la nouvelle position et la nouvelle direction visuelle que l'on souhaite obtenir, en cherchant à minimiser l'erreur.

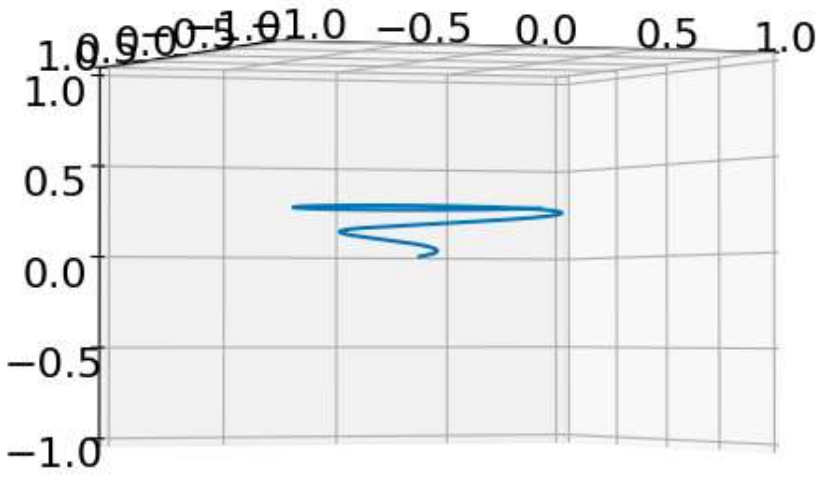

FIGURE 2.9 – Exemple de trajectoire de l'effecteur en 3D pour un test de suivi de direction visuelle.

La figure 2.14 montre un exemple de trajectoire de l'effecteur du bras quand on lui demande de suivre une direction visuelle constante. On remarque que la trajectoire n'est pas directe mais hélicoïdale mais que le bras arrive à se positionner correctement et à

effectuer des ellipses tout en gardant l'effecteur sur une ligne droite dans le domaine visuel.

2.8 Autres réseaux

J'ai développé de nombreux modèles différents au cours de mon stage. En effet, le modèle que j'utilise étant assez général, il a fallut faire un choix sur de nombreux hyperparamètres (taille et nombre des filtres, taille des vecteurs de sortie, etc.) J'ai listé ci-après quelques remarques sur ces modèles.

1. Multiplication tensorielle ou multiplication terme à terme ? Utiliser une multiplication terme à terme comme évoqué dans la section sur la factorisation du tenseur a permis de diminuer le temps d'apprentissage sans pour autant réduire la performance du réseau. On remarque que la multiplication tensorielle n'est pas adaptée à un exemple comme le nôtre et qu'il contient beaucoup trop de paramètres à optimiser pour la difficulté du problème considéré.
2. Posture ou image ? Par posture j'entends la donnée des angles faits par les articulations. En donnant des pairs de posture plutôt que des pairs d'images en entrée du premier réseau, j'ai observé une réduction du temps d'apprentissage, en effet, le réseau a une information directe et sûre concernant la position du bras et n'a plus besoin de l'inférer à partir des images.
3. Direction visuelle ou image ? J'ai utilisé également dans un premier temps, les directions visuelles associées aux mouvements effectués par le bras robotique en calculant le vecteur de direction associé.

2.9 armcodlib

On peut retrouver tous mes expériences et codes sur `https://github.com/elichou/notebooks_gfn` ce qui comprend :

1. la modélisation du bras fil de fer en Python.
2. le contrôle du bras fil de fer.
3. l'implémentation des modèles avec tensorflow et keras.
4. l'implémentation des modèles de factorisation.
5. l'implémentation des analyses de filtres convolutifs.
6. l'implémentation des analyses de champs réceptifs.
7. l'utilisation de l'algorithme t-sne.
8. un package ROS pour la création d'une base de données pour l'apprentissage.
9. un package ROS pour le contrôle du bras robotique avec des modèles en Python.

10. une amélioration du package kinova-ros avec une correction de l'URDF du bras robotique.
11. toutes les autres fonctions qui composent le fichier armcoblib.py.

2.10 Analyse de quelques résultats

Il s'agit de résultats partiels qui feront l'objet d'une étude plus approfondie dans l'article qui sera soumis à publication dans l'édition "Computational models of affordances for robotics" dans la revue "Frontiers in Neurorobotics".

2.10.1 Algorithme t-snE

Quand on fait passer une pair d'images dans le premier réseau, on obtient en sortie un vecteur (typiquement de taille 32 ou 64), (voir figure 2.10) qui n'est pas interprétable directement. J'ai utilisé un algorithme t-sne qui est un algorithme de réduction de dimensionnalité, qui permet de représenter visuellement en 2D ou en 3D des données de grande dimension. Sur la figure 2.11 on peut observer le résultat de cet algorithme quand on passe d'un vecteur de sortie à une projection sur deux dimensions uniquement. Chaque couleur représente l'activation maximale pour un neurone du vecteur de sortie. Ce n'est pas forcément un bon critère puisqu'on a un codage en population et que le résultat est un mixte des activations des neurones mais cela permet d'observer tout de même cette répartition en population. Cet algorithme minimise la divergence de Kullback-Leibler entre la distribution de points dans l'espace de grande dimension et la répartition des points dans l'espace 2D, c'est une méthode probabiliste.

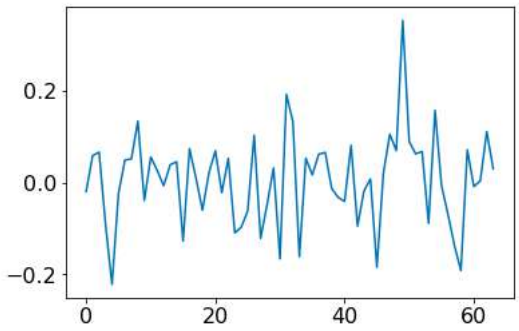

FIGURE 2.10 – Exemple de vecteur de sortie du premier réseau. En abscisse le numéro du neurone, en ordonnée son activation.

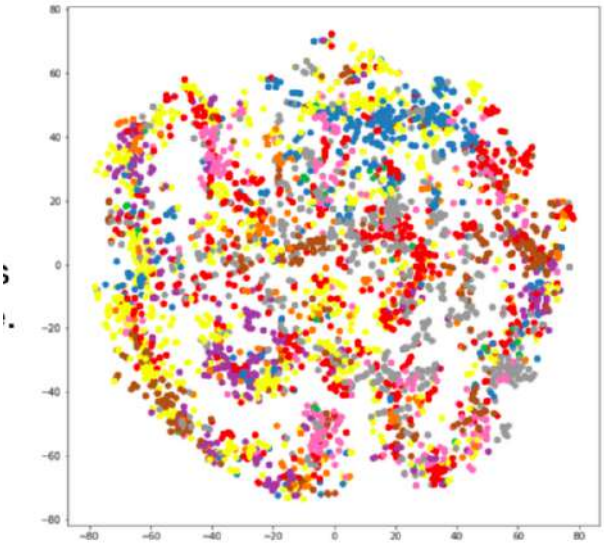

FIGURE 2.11 – Algorithme t-sne appliqué aux sorties du premier réseau.

2.10.2 Filtres convolutifs

La figure 2.12 montre les filtres convolutifs appris par le premier réseau. Pour obtenir ces filtres, on réalise une descente de gradient afin d'obtenir les images qui vont maximiser la valeur des neurones de la couche de factorisation des entrées. On peut remarquer d'une part que ces filtres vont être soit localisés uniquement à certains endroits de l'image, soit spécifiques à des directions particulières des mouvements. On s'attendait à voir des filtres qui indiquait une notion de profondeur avec une sélectivité pour des mouvements allant vers la caméra ce qui n'a pas été le cas.

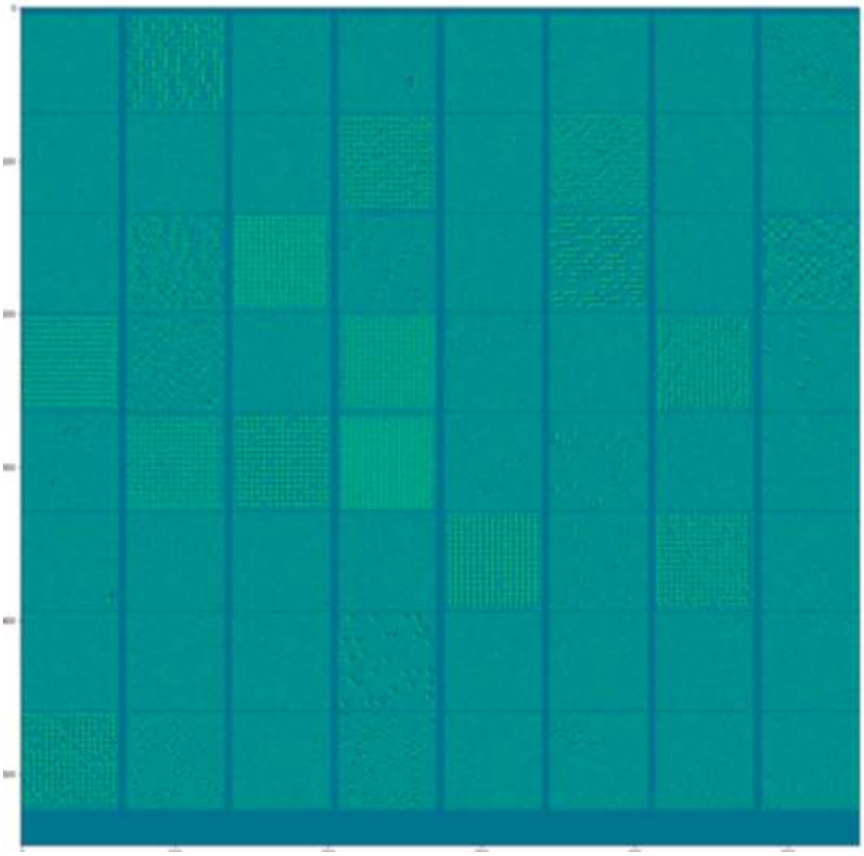

FIGURE 2.12 – Filtres convolutifs de l'encodeur

2.10.3 Champs réceptifs

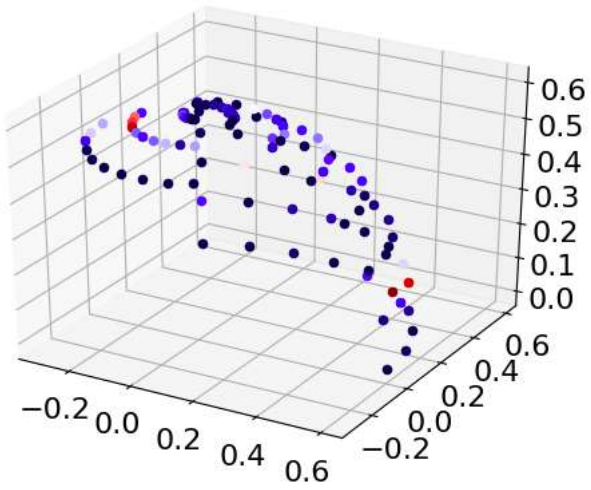

FIGURE 2.13 – Activation d'un neurone en fonction de la position initiale de l'effecteur (en couleur).

La figure 2.13 montre l'activation d'un neurone en fonction de l'endroit d'où est réalisé un mouvement. Le rouge représente une activation positive tandis que le bleu représente une activation négative. on observe que ce neurone ne va s'activer que quand le mouvement commence autour des deux lieux de l'espace de travail où l'activation est positive. Il est ainsi intéressant de remarquer que certains neurones présentes des lieux d'activation spécifique, comme les neurones prémoteur qui présente des directions préférentielles dans leur activation. Pour autant, c'est bien un mixte de ces neurones qui présentent une activité pertinente pour le calcul d'une commande motrice.

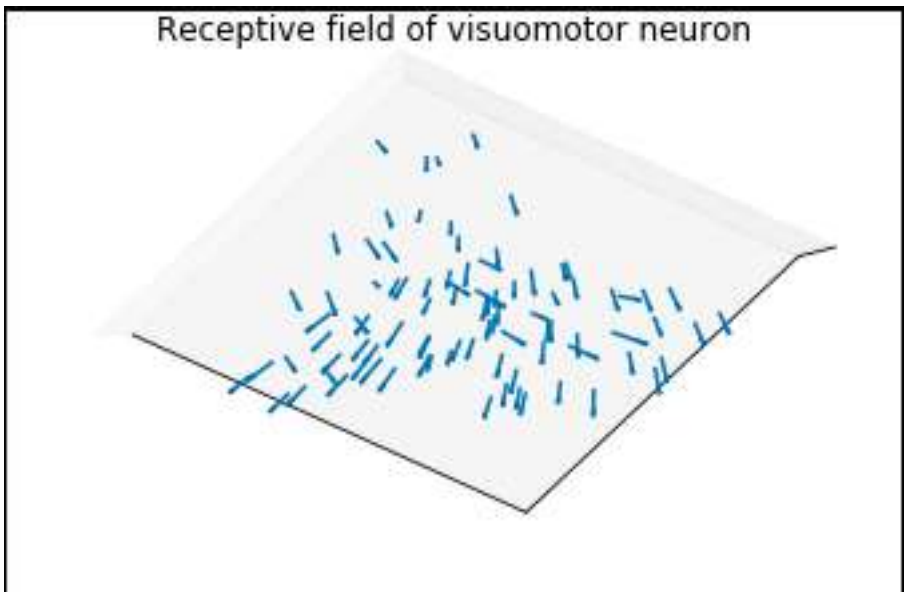

FIGURE 2.14 – Vue du dessus du champ réceptif d'un neurone

La figure 2.14 présente le champ réceptif d'un neurone particulier de la couche de factorisation. Pour un certain nombre de lieux de l'espace de travail, on cherche quelle est la direction du mouvement à effectuer qui va maximiser l'activation d'un neurone. On remarque ici qu'on a deux directions privilégiées (vers la droite et vers la gauche). On a donc pas une direction privilégiée pour tous l'espace mais plutôt des directions privilégiées localement.

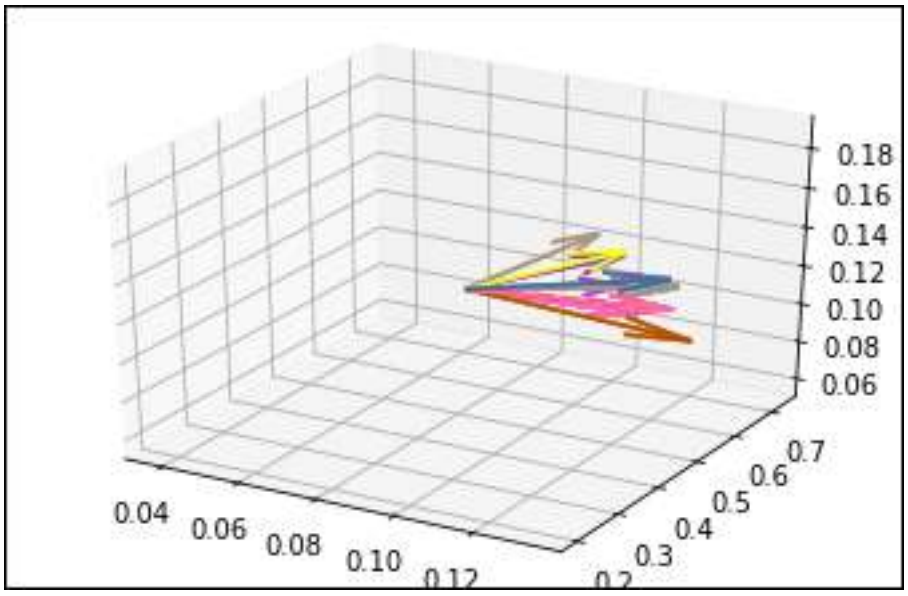

FIGURE 2.15 – Contribution de chaque neurone

Sur la figure 2.15 on observe les différentes contributions de chaque neurone pour un mouvement donné, c'est à dire qu'on regarde quelle commande est induite par chacun des neurones. On remarque qu'on a un codage des commandes en population c'est à dire qu'on a pas un neurone spécifique à une direction mais c'est la combinaison des différentes contribution qui amène à la commande correcte.

2.11 Simulation ROS/Gazebo

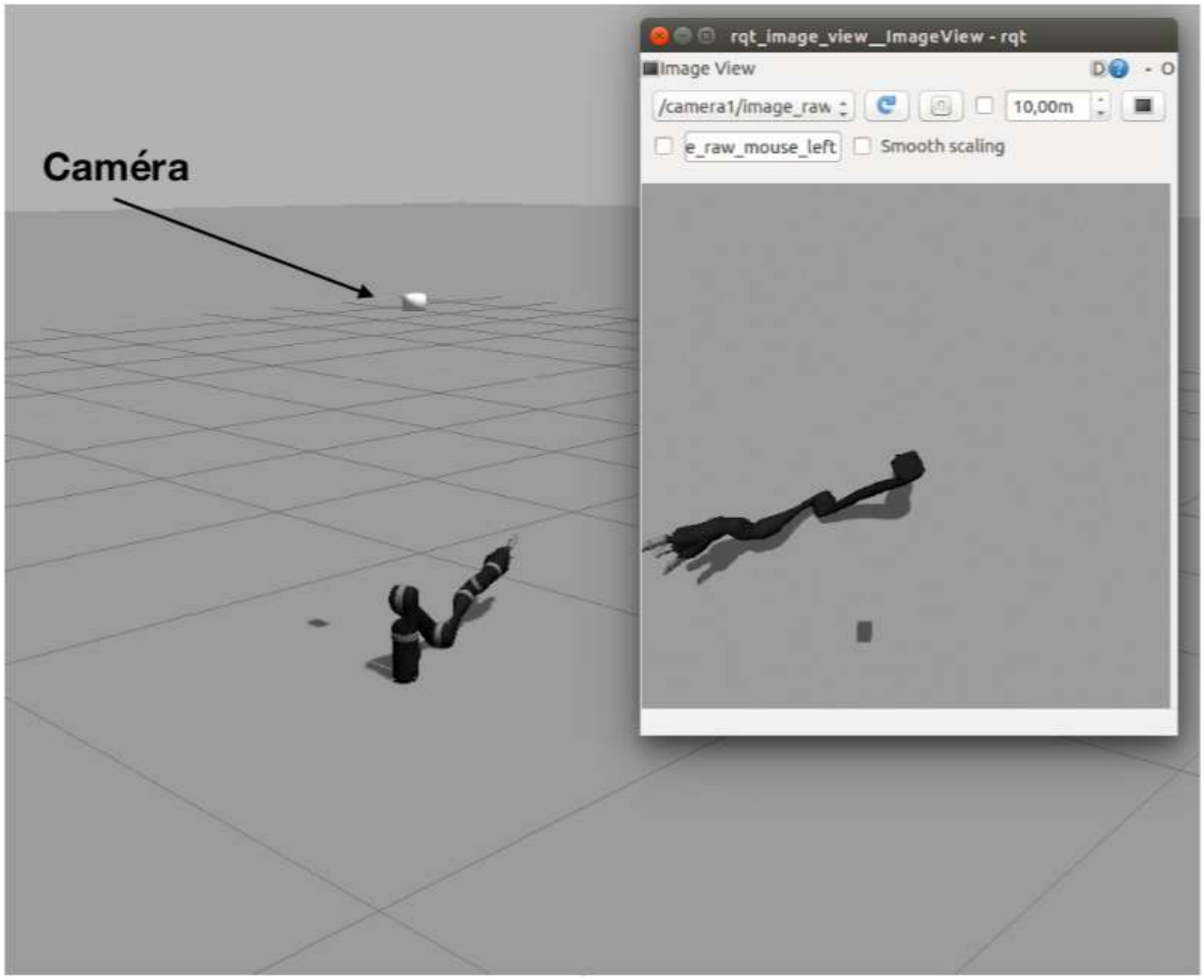

FIGURE 2.16 – Setup du bras robotique utilisé pour tester les modèles.

J'ai développé des packages ROS pour la création d'une base de données ainsi que pour le contrôle du bras de chez Kinova Robotics à partir de leur package kinova-ros. Cette

implémentaton n'a pas été de tout repos du fait de problèmes de compatibilité entre les versions de ROS, d'Ubuntu, de Gazebo et d'un module de calcul de cinématique inverse et de la clarté des erreurs de ROS. Ces expériences sont actuellement en cours et n'ont pas encore données de résultats mais feront partie de l'article soumis à publication.

Conclusion et perspectives

L'utilisation des neurosciences pour le développement de la robotique est un domaine extraordinairement riche, à la croisée entre l'informatique, la robotique et les neurosciences. Le développement de modèle de contrôle visuo-moteur en utilisant le mécanisme de modulation en gain ("gain-field modulation") a permis de montrer qu'il était possible d'apprendre des affordances liant directement les effets captés par les senseurs aux actions réalisées par les actionneurs. Dans ce projet j'ai développé un "convolutional gated auto-encoder" pour le contrôle visuo-moteur en robotique. Les résultats expérimentaux sont encourageants puisqu'ils ont permis de retrouver des résultats obtenus en neurosciences sur le fonctionnement du cortex pariétal. Malgré cela, il manque une approche systématique pour le choix des hyperparamètres des modèles ainsi qu'un cadre qui permettrait de comparer plus facilement les résultats obtenus. Ces travaux vont faire l'objet d'un article qui sera soumis à la publication dans le journal "Computational models for affordances in Robotics" en Octobre 2019. J'ai le regret de ne pas pouvoir continuer à travailler sur ce sujet qui me semble très prometteur et très riche.

Bibliographie

[1] Julien Abrossimoff, Alexandre Pitti, Philippe Gaussier, *Visuo-motor control from generated body images using gated auto-encoders*, Université Paris-Seine, Laboratoire ETIS, UMR 8051, 2015.

[2] Ganna Pugach, Alexandre Pitti, Olga Tolochko, Philippe Gaussier, *Brain-inspired coding of robot body schema through visuo-motor integration of touched events*, Frontiers in Neurorobotics, March 2019.

[3] Julien Abrossimoff, Alexandre Pitti, Philippe Gaussier, *Visual learning for reaching and body-schema with gain-field networks*, 2018 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics, Tokyo, Japan, Septembre 2018.

[4] Akira Murata, Wen Wen, Hajime Asama, *The body and objects representation in the ventral stream of parieto-premotor network*, Neuroscience Researhc, Elsevier, Mars 2016.

[5] Apostolos Georgopoulos, Costas Stefanis, *Local shaping of function in the motor cortex : motor contrast, directional tuning*, Brain Research Review, Elsevier, Mai 2007.

[6] S. Kakei et al., *Muscle and movement representations in the primary motor cortex*, Science. 1999 Sep 24 ;285(5436) :2136-9.

[7] Baraduc, Emmanuel Guigon, Yves Burnod, *Recoding Arm Position to Learn Visuomotor Transformations*, Cerebral Cortex, Volume 11, Issue 10, October 2001, Pages 906–917.

[8] Olivier Sigaud, Clément Masson, David Filliat, Freek Stulp, *Gated networks : an inventory*, non publié, 2016, hal-01313601.

[9] G. Blohm, A. Z. Khan, J. D. Crawford, *Spatial transformations for eye-hand coordination*, University of Toronto, Elsevier, 2008.

[10] Roland Memisevic, *Learning to relate images*, Univeristy of Montreal, 2007.

[11] Roland Memisevic, K. Konda, *Unsupervised learning od depth and motion*, arXiv :1312.3429v2 , 2013.

[12] Roland Memisevic, H. Kamyshanska, *On autoencoder scoring*, International Conference on Machine Learning, 2013.

[13] Roland Memisevic, H. Kamyshanska, *The potential energy of an autoencoder*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

[14] Hideo Sakata *et al.*, *Neural representation of three-dimensional features of manipulation objects with stereopsis*, Exp Brain Research, Mars 1999.

[15] Alexandre Pouget, Terrence J. Sejnowski, *Spatial Transformations in the parietal cortex using basis functions*, Journal of Cognitive Neuroscience, 9 :2, pp. 222-237, MIT, 1997.

[16] Aleksi Hamalainen et al., *Affordance learning for end-to-end visuomotor robot control*, KTH, Mars 2019.

[17] Mihai Andries et al., *Affordance equivalences in robotics : a formalism*, Frontiers in Neurorobotics, Juin 2018.

[18] Srinivas Chivukula et al., *Cognition in sensorimotor control : interfacing with the posterior parietal corex*, Frontiers in Neuroscience, Février 2019.

[19] Luis Montesano et al., *Learning object affordances : from sensory-motor coordination to imitation*, IEEE Transactions On Robotics, Vol. 24, n1, Février 2008.

[20] James Greeno, *Gibson's affordances*, Psychological Review 1994, Vol.101,No. 2, 336-342.

Abstract — Learning to link actions with objects, what we call affordances, is still a difficult task for robots. It requires not only to learn how the body moves with respect to its morphology but to learn also how it is relatively to moving objects with different orientations and positions with respect to it. The parietal cortex is responsible for encoding object/body spatial relationships by integrating multimodal information. In effect, it codes the relative distance and orientation that allow to represent objects in relative coordinates centered on the body parts in order to sense them and to reach them. The mechanism for binding visual, proprioceptive, tactile and motor information is the so- called gain modulation mechanism, which implements sensorimotor transformations from global coordinates to relative coordinates (e.g. hand-centred coordinates in visual system) from intertwinned representations. From a computationl viewpoint, these mixted representations are interesting as they can serve then to generate forward and inverse models in various spatial reference frames. In this report, we propose to model these features in three architectures and robotic experiments in order to achieve the learning of 3D motor cells, the encoding of 3D visual motion, and the alignment between hand and objects toward 3D reach with proprioceptive, motor and visual integration exploiting gain-modulation. In a first experiment with a multi-degrees of freedom robot, we propose to model neurons the motor visual direction cells found by Georgopoulos in the 1980's using the gain-modulation mechanism. These motor neurons are sensitive to visual directions and responsible for 3D reaches and the learning of motor primitives. In a second experiment using visual and motor information, we will develop a neural network that will exploits mixed representation for learning the affine visual transformations relative to 3D motion in space. This property found in the parietal cortex by Kakei is particularly interesting to detect 3D motion of objects in space and of the hand ; this network is important in order to control movement and select actions further. In a third experiment, we will develop a third network that combines the capabilities of the two previous ones for controlling and aligning the robot hand to particular 3D orientations with respect to a particular target location and orientation. **Mots clés :** learning, neurorobotics, affordances, reaching, gated auto-encoders

ENSTA Bretagne
2 rue François verny
29200 Brest