Utilisation des réseaux de neurones récurrents à temps continu dans le contexte d'une interaction sensorimotrice temps réel minimaliste crédible

Roman Culioli

Master 2 Recherche Informatique,
2012-2013

Rapport de stage

Utilisation des réseaux de neurones récurrents à temps continu dans le contexte d'une interaction sensorimotrice temps réel minimaliste crédible.

Auteur:
Roman Culioli

Superviseurs:
Pierre De Loor
Alexis Nédélec

Equipe:
IHSEV

Abstract

Ce stage aspire à utiliser une architecture cognitive afin de créer un programme capable d'interagir gestuellement en 2D avec un humain et de se faire passer pour tel. Le but étant de comprendre les mécanismes d'obtention de la crédibilité par les agents de réalité virtuelle lors d'interactions gestuelles minimalistes. Une étude des différentes architectures existantes nous permet de retenir les CTRNN comme étant la plus prometteuse. On paramètre ceux-ci à l'aide d'algorithmes génétiques, en visant à promouvoir deux aspects : le réalisme des mouvements et la qualité de l'interaction avec l'humain. Le programme est ensuite testé au cours d'expérimentations. L'exploitation des résultats à la fois pour évaluer le protocole et le sentiment de crédibilité obtenu par nos agents est ensuite présentée.

Table des matières

1 Introduction **4**

2 Travaux précédents **5**

3 État de l'art **7**
3.1 Architectures symboliques 7
3.2 Architectures émergentistes 7
3.2.1 Principes généraux 7
3.2.2 Exemple d'architecture : CTRNN 8
3.3 Architectures hybrides 9
3.3.1 Principes généraux 10
3.3.2 Exemple d'architecture : OCP 11
3.4 Architectures pour la robotique 12
3.4.1 Principes généraux 12
3.4.2 Exemple d'architecture : EICA 13
3.5 Choix de l'architecture 14

4 Évolution de la configuration du CTRNN **15**
4.1 Les pièges de l'approche évolutionnaire 16
4.2 Réalisme des mouvements 18
4.3 Prise en compte de la loi de Viviani 20
4.4 Début d'interaction : suivi de trajectoire 23
4.5 Corrélation des trajectoires : Cross Correlation 24
4.6 Version finale de la fonction d'évaluation 27
4.7 Enregistrement des réseaux 27

5 Les expérimentations **29**
5.1 Les différents algorithmes 30
5.1.1 Miroir 30
5.1.2 Aléatoire 30
5.1.3 Enregistrement 30
5.1.4 Enregistrement sans collision 31
5.1.5 CTRNN 31
5.2 Les questions 31
5.3 Résultats 32
5.3.1 Questionnaire numérique 33
5.3.2 Temps de détection 35
5.3.3 Questionnaire post-expérimentation 35

6 Conclusion et perspectives **36**

Table des figures

1 Interface 2D 6
2 Etude graphique des points de récurrence dans l'interaction entre deux humains. 6
3 Topologie d'un CTRNN. 9
4 Diagramme haut niveau de l'architecture CogBot. 11
5 Embodied Interactive Control Architecture (EICA). 13
6 Apprentissage sur un seul enregistrement. 17
7 Apprentissage de la forme. 18
8 Fonction de fitness trop simple. 18
9 Trajectoire contrôlée en vitesse et en rotation. 19
10 Comportement respectant des contraintes de vitesse et de rotation. 20
11 Courbe 3D de la loi de Viviani sur un tracé humain, vue sous trois angles différents. 22
12 CTRNN avec application de la loi de viviani. 23
13 Trajectoire sous contrainte de distance avec l'humain. 23
14 Corrélation en position. 24
15 Position en X (vert) et Y (rouge). 25
16 Corrélation en position en X et Y. 25
17 Vitesse en X (vert) et Y (rouge). 26
18 Corrélation en vitesse. 26
19 Trajectoire finale du réseau de neurone. 27
20 Structure du réseau de neurones. 28
21 Fichier formaté pour graphviz. 28
22 Organisation de l'expérimentation. 29
23 Interface du questionnaire. 32
24 Question 1 : Sentiment de présence. 33
25 Question 2 : Cherche l'interaction. 33
26 Question 3 : Humain ou machine. 34
27 Question 4 : Prise en compte de l'autre. 34
28 Question 5 : Trajectoire prédéfinie. 35

1 Introduction

Ce stage s'inscrit dans le projet INGREDIBLE (INteraction Gestuelle cREDIBLE) financé par l'Agence Nationale de la Recherche. Ce projet, mené par l'équipe IHSEV (Interaction Humain Système et Environnement Virtuel) du laboratoire Lab-STICC, a pour objectif de créer un acteur virtuel autonome qui procurera un ressenti de "présence" de l'autre aux utilisateurs de systèmes de réalité virtuelle du futur.
Construire un programme d'intelligence artificielle ne pouvant être différencié d'un véritable être humain par un autre être humain interagissant avec lui est le défi qu'a posé Alan Turing en 1950. Dans son article *Computing machinery and intelligence* il décrit un test permettant de juger la crédibilité d'un tel programme. Et c'est une variante du test de Turing, le test de Turing sensorimoteur, que le projet INGREDIBLE contribue à satisfaire. Le but est donc de reproduire le couplage dynamique présent lors d'une interaction réelle entre des humains. Ce couplage dynamique se manifeste par le fait que chaque action d'un des interlocuteurs modifie le comportement de l'autre interlocuteur et réciproquement. Ce couplage entraîne une coordination des acteurs, comportant des phases de régularité, *d'imitation*, et des phases aléatoires, *d'initiatives*. Ce projet ne traite que la partie gestuelle du couplage ignorant donc les aspects verbaux et linguistiques.
L'objectif du stage de master consistera à choisir et à mettre en place une architecture cognitive fournissant à un agent virtuel une capacité de couplage dynamique corporel avec un humain durant une interaction minimaliste n'impliquant qu'une partie du corps (la main).
Selon un article de P.De Loor[Tis11] les systèmes dynamiques possèdent des propriétés particulièrement intéressantes pour atteindre cet objectif. Tout d'abord l'aspect très complexe et quasi imprévisible sont des notions que partagent systèmes dynamiques et couplage sensorimoteur. Les systèmes dynamiques couplés sont fortement sensibles aux perturbations et peuvent faire apparaître des attracteurs collectifs. Ces récurrences donnent des tendances comportementales à ces systèmes qui peuvent se rapprocher des comportements sensorimoteurs biologiques. Selon les auteurs de cet article les modèles actuels basés sur les statistiques ne peuvent pas fonctionner pour notre problème. Car en réduisant l'ensemble des comportements à des probabilités on rend non seulement l'agent prévisible mais on définit également des limites qui, dès qu'elles seront franchies, démasqueront le programme à coup sûr. L'agent doit donc se construire lui même une mémoire à partir des invariants sensorimoteurs, ce qui est la base de la constitution du sens suivant le paradigme énactiviste. Cette mémoire permet à l'agent de générer autre chose que de l'aléatoire et donc d'obtenir un équilibre entre surprise et régularité.

Cet équilibre est essentiel pour garantir la crédibilité du programme.
Il existe de nombreuses architectures cognitives mais peu font appel aux notions précédemment citées. Récemment un nouveau type d'architecture cognitive est apparu. Ces architectures cognitives "bio-inspirées" sont généralement issues des recherches en neurosciences et en sciences cognitives. Elles intègrent des éléments de complexité du fonctionnement cognitif et paraissent plus aptes à reproduire les comportements sensorimoteurs complexes des humains. Je concentrerai donc mes recherches autour de ces architectures.
Dans un premier temps je m'appuierai sur les travaux d'un stage précédent afin de déterminer les critères qui me permettront de choisir l'architecture la plus adaptée. Je choisirai celle-ci à l'issue de la troisième partie dans laquelle j'étudie les différentes architectures cognitives existante. La quatrième partie concernera l'évolution de la configuration de notre architecture cognitive jusqu'aux expérimentations, elles mêmes détaillées dans la cinquième partie.

2 Travaux précédents

Ce stage s'inscrit dans la continuité d'un autre stage [Val12] effectué l'année dernière dans le cadre du master de recherche informatique. Ce stage a permis de mener une étude à la fois quantitative et qualitative de la crédibilité des interactions humain/humain et humain/programme. Visant à déterminer les critères fondamentaux assurant cette crédibilité, T.Vallée s'est limité à l'étude d'interactions minimalistes dans laquelle les protagonistes ne sont représentés que par un cercle se déplaçant dans un environnement 2D (Figure 1). Chacun des cercles est contrôlé soit par un humain soit par un programme et la seule consigne donnée aux utilisateurs est de deviner si ils interagissent avec un autre humain ou avec un programme.

Cette étude met en évidence le fait qu'utiliser une échelle de temps différente de celle du temps de réaction d'un humain améliore la capacité de l'utilisateur à détecter le programme. La réponse en temps réel sera donc un des critères de choix de l'architecture cognitive. Cette étude montre également la présence de phénomènes de récurrence dans le couplage sensorimoteur grâce aux graphiques de points de récurrence (Figure 2). Une RQA (*Recurrence quantification analysis*) relève sur un graphe les instants où durant une interaction il y a de fortes corrélations sur un signal. Ainsi, ici abscisses et ordonées représentent le temps et la teinte des points en x_i, y_i représente le fait que la souris était dans une zone proche au temps x_i et au temps y_i. Plus le graphique comporte de "tâches", plus les points de récurrence sont nombreux. Un programme ne possédant pas de récurrence donc purement aléatoire (aucune "tâche") est rapidement identifié. De la

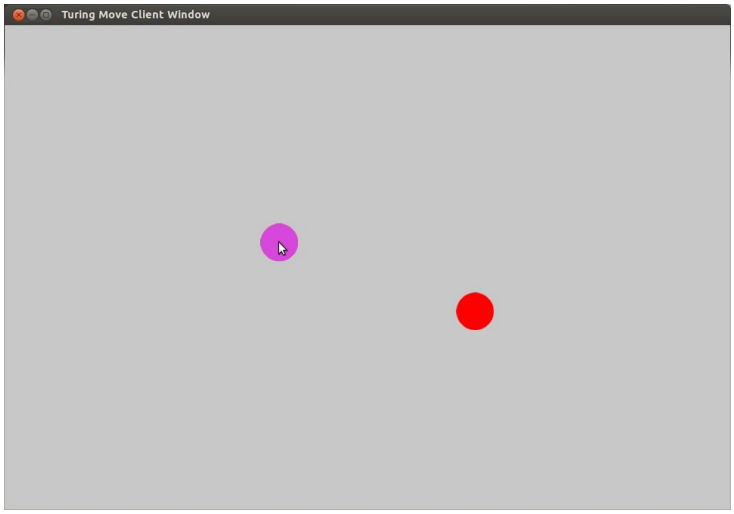

FIGURE 1 – Interface 2D

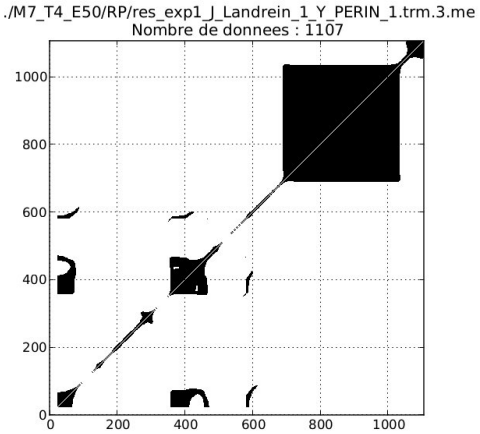

FIGURE 2 – Etude graphique des points de récurrence dans l'interaction entre deux humains.

même façon, un programme en possédant trop n'est pas crédible car trop prévisible. Cela souligne une caractéristique déjà identifiée par [Ike09], l'équilibre entre l'aspect d'initiative ou d'autonomie et l'aspect de régularité de l'interaction entre deux humains.

Les critères de choix d'architecture pour la suite de l'étude seront donc :

- Interaction temps réel
- Équilibre entre autonomie et synchronie
- Pas de connaissance pré-requise

Pour structurer ma recherche je me suis appuyé sur des articles qui passent en revue les différents types d'architectures cognitives. D.Vernon

[San07] présente une analyse comparative des différents paradigmes de cognition, suivie d'une présentation d'une panoplie d'architectures cognitives classées en trois groupes : les architectures cognitivistes ou symboliques, les architectures émergentistes et les architectures hybrides. A ces trois catégories B.Goertzel [dGSC10] vient en ajouter une quatrième, les architectures cognitives pour la robotique.

3 État de l'art

3.1 Architectures symboliques

L'approche symbolique est la méthode classique pour la construction d'architectures cognitives. Les plus célèbres architectures cognitives comme ACT-R ou Soar font partie de cette catégorie et ont fait leurs preuves quant à leur efficacité à effectuer des raisonnements logiques. Mais dans notre cas, les bases même de ce paradigme posent problème. Les informations étant représentées de façon symbolique, de manière à être accessible à une interprétation humaine directe, sont d'un niveau d'abstraction trop élevé pour représenter la nature du couplage sensorimoteur. Les architectures symboliques traitent l'information de façon structurée et algorithmique en se basant sur des règles et des probabilités, ce qui est incompatible avec notre problème. Aussi, les notions d'autonomie et d'*embodiment* qui sont essentielles pour le projet ne sont pas particulièrement présentes dans les fondements de ce paradigme. C'est pour ces raisons que j'ai choisi d'exclure toutes les architectures de cette catégorie.

3.2 Architectures émergentistes

Les architectures émergentistes englobent trois types de système [San07], les systèmes dynamiques, les systèmes connexionnistes et les systèmes énactifs. Ils reposent tous sur des principes d'auto-organisation. C'est de cette auto-organisation qu'émerge de nouveaux comportements.

3.2.1 Principes généraux

L'intérêt des systèmes dynamiques a déjà été mentionné (instabilité et attracteur). Les systèmes connexionnistes basés sur les réseaux de neurones peuvent être assimilés, dans certains cas, à des systèmes dynamiques et sont bien connus pour leur capacité de classification, de détection des régularités et pour leur faculté d'apprentissage. Ils peuvent jouer le rôle de mémoire grâce

aux rétroactions. Et sont parfaitement aptes à recevoir les données sensorielles de notre futur agent (contrairement aux architectures symboliques). Enfin, suivant les principes de l'énaction, les invariants sensorimoteurs sont la base de la création du sens chez les êtres vivants [Tis11]. Les systèmes énactifs créent leur propre représentation du monde, il n'y a donc rien de prédéfini. Et c'est un point important étant donné la complexité à définir l'ensemble des comportements lors des interactions entre humains. La co-évolution est également une notion fondamentale du paradigme énactif [Tis09].
J'ai choisi de détailler une architecture émergentiste parmi les principales existantes : Global Workspace, I-C SDAL, SASE, DARWIN, HTM, DESTIN, NOMAD, CTRNN. J'ai fait mon choix par élimination, suivant la description de ces architectures dans les articles [San07] et [dGSC10]. L'architecture HTM est exclusivement utilisée pour le traitement de la vision. NOMAD fait surtout de la classification. Global Workspace est structuré en de nombreux processus fonctionnant en parallèle et ne semble donc pas utilisable pour notre projet. Les architectures I-C SDAL, DARWIN et DESTIN sont basées sur l'apprentissage par renforcement, ce qui est difficilement applicable dans notre cas car, pour l'instant, on ne peut pas mesurer la crédibilité en temps réel. L'architecture SASE est toujours à un stade théorique et ne possède actuellement aucune implémentation. Les CTRNN possèdent quant à eux de nombreux aspects intéressants que nous allons détailler maintenant.

3.2.2 Exemple d'architecture : CTRNN

Les réseaux de neurones récurrents à temps continu (CTRNN) sont un type de réseaux de neurones artificiels caractérisés par une topologie qui autorise toutes les connexions et par le fait que chaque neurone possède sa propre dynamique [Man11]. Tous les neurones sont reliés entre eux par des liaisons pondérées et ils possèdent également une liaison pondérée vers eux même. L'organisation de ce type de réseau est circulaire, il n'est pas possible de distinguer des couches successives de traitement de l'information. La figure (3) montre un exemple de CTRNN, certains neurones reçoivent une entrée extérieure et la sortie de certains neurones est considérée comme sortie du réseau.

L'évolution de l'état d'activation des neurones dans le temps est régie par l'équation différentielle suivante :

$$\tau_i \frac{\delta y_i}{\delta t} = -y_i + \sum_{j=1}^{k} w_{j,i}\sigma_j + I_i \tag{1}$$

Avec τ une constante de temps, y_i l'activation du neurone i, k le nombre total de neurones, $w_{j,i}$ le poids de l'arc (j,i), σ_j l'activation du neurone j modifié

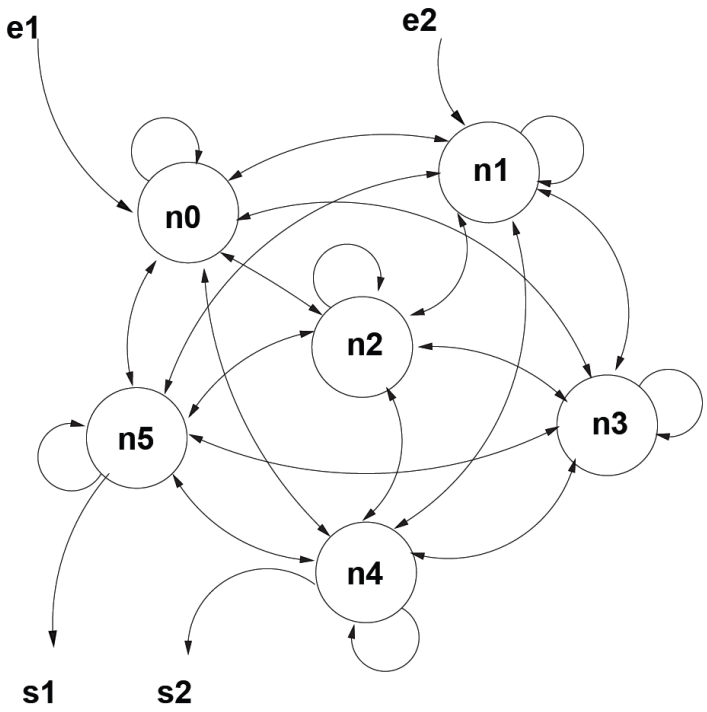

FIGURE 3 – Topologie d'un CTRNN.

pour suivre une forme de sigmoïde et I_i la valeur de l'entrée appliquée au neurone i.

L'équation de l'activation des neurones suivant une forme de sigmoïde :

$$\sigma_i = \frac{1}{1 + e^{-(y_i + b_i)}} \tag{2}$$

Avec y_i l'activation du neurone i et b_i le biais.

Chaque neurone ayant sa propre constante de temps τ, ils évoluent selon une fréquence spécifique. Dans un même réseau, il peut donc y avoir des neurones qui réagissent de manière quasi réactive à leurs entrées alors que d'autres auront une grande inertie. La dynamique globale du réseau peut donc refléter des dynamiques opérant à différentes échelles de temps. Les CTRNN présentent l'avantage théorique de pouvoir reproduire le comportement de n'importe quel système dynamique.

3.3 Architectures hybrides

Les architectures hybrides combinent les approches cognitivistes et émergentistes dans le but d'allier les forces et ainsi diminuer les faiblesses de ces architectures.

3.3.1 Principes généraux

Tout d'abord, les principaux défauts que les systèmes émergentistes sont aptes à corriger chez les systèmes cognitivistes sont l'ancrage symbolique et l'explosion combinatoire, défauts qui rendent difficile les interactions sensorimotrices dans des environnements dynamiques complexes. Ensuite, les modèles cognitivistes ont des difficultés à créer, à généraliser, à apprendre et à fonctionner dans des domaines qui ne sont pas complètement définis. Or ce sont les points forts des modèles émergentistes. Mais un système dynamique n'est pas en soi un moyen de traiter de l'information. Et selon [San07],les capacités actuelles à produire des systèmes exclusivement non-symboliques sont pour le moment très limitées car les théories décrivent plus des principes généraux que de véritables modèles cognitifs implémentables. De ce point de vue, les architectures cognitivistes sont bien plus avancées et ont déjà montré de sérieuses capacités à traiter des problèmes du domaine du raisonnement et de la compréhension linguistique par exemple. En combinant les deux paradigmes, les agents d'architectures hybrides peuvent donc posséder des pré-requis, définis par leur système de représentation symbolique, et s'adapter à leur environnement, au travers de leur expérience et leur apprentissage. Les architectures hybrides sont donc conformes aux principes des systèmes émergents et les calculs non-symboliques de ces systèmes viennent compléter la représentation symbolique des systèmes cognitivistes. La partie émergentiste s'occupe généralement des connaissances implicites alors que la partie symbolique gère les connaissances explicites. Un système hybride est usuellement composé de multiples sous-systèmes possédant des fonctions indépendantes, fonctionnant selon l'un des deux paradigmes et communiquant leurs résultats entre eux. Les opinions concernant ces architectures sont divisées compte tenu de la nature profondément antagoniste des deux paradigmes combinés. Les principales architectures hybrides sont CLARION, DUAL, LIDA, MicroPsi, PolyScheme, Shruti et OpenCogPrime. J'ai choisi d'éliminer Shruti car, selon [dGSC10], après un long temps de développement cette architecture n'a montrée que très peu de résultat. PolyScheme est focalisée sur la résolution de problème. L'architecture MicroPsi est basée sur la motivation et la satisfaction de besoin, elle est donc difficilement applicable dans notre cas. CLARION et DUAL sont exclues car elles ne sont pas encore tout a fait abouties. J'ai donc choisi de détailler l'architecture OCP qui selon son créateur est une instanciation de LIDA.

3.3.2 Exemple d'architecture : OCP

L'architecture OpenCogPrime (OCP) a été créée par Goertzel [Pen10] pour le contrôle d'agent virtuel intelligent. Il fait partie du framework OpenCog, qui est un projet open-source. Le but de cette architecture est de permettre à l'agent virtuel intelligent d'exécuter la procédure qui lui paraît la plus adaptée en fonction de ces objectifs et du contexte courant. On peut voir sur l'organisation générale d'OCP (Figure 5) que la mémoire est di-

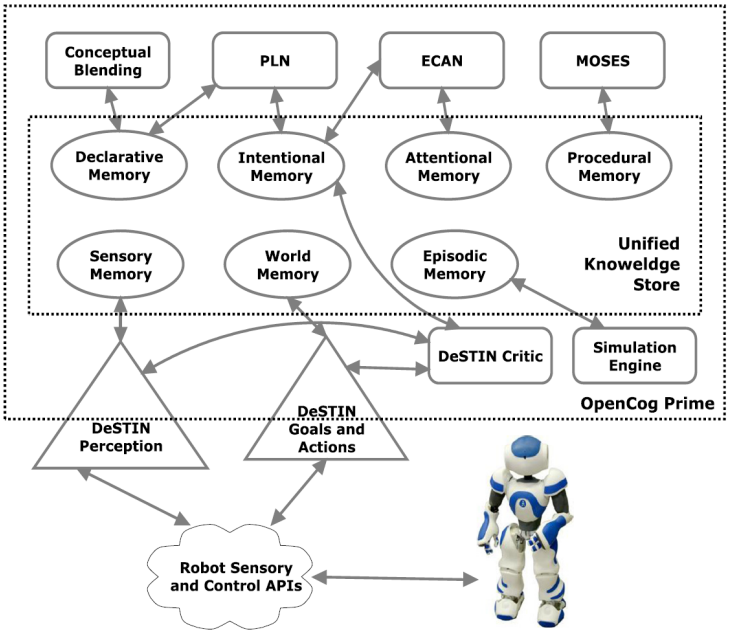

FIGURE 4 – Diagramme haut niveau de l'architecture CogBot.

visée en plusieurs types différents (déclarative, intentionnelle, attentionnelle, procédurale, épisodique et sensorielle), chacun associé à des processus cognitifs. Cette disposition de la mémoire permet le stockage de modèles particuliers de façon efficace et optimisée. Les mémoires sont conçues pour collaborer de façon étroite et engendrent une intelligence coopérative, allant plus loin que les architectures de structure similaire composées de "boites noires" séparées. Goertzel explique dans cet article [Goe09] que cette coopération de la mémoire crée une synergie cognitive. OCP fonctionne à la fois de façon *goal-oriented* et de façon spontanée, une application de cette architecture a déjà montré des fonctionnalités intéressantes pour notre problème : l'agent était capable d'apprendre de nouveaux comportements par imitation et répondait au langage naturel par du langage naturel. Mais en installant OCP je me suis rendu compte que cette architecture était beaucoup trop développée pour

gérer le seul mouvement de base de notre application qui est le déplacement du curseur de la souris. Plus de 90% des modules seraient restés vide. OCP demande plus de connaissances que nous en possédons sur le système que nous voulons simuler. OCP étant donc un outil trop puissant pour manier l'interaction minimaliste de notre application, nous avons décidé de ne pas l'utiliser.

3.4 Architectures pour la robotique

L'usage croissant de robot intégrant une notion d'intelligence artificielle a amené beaucoup de chercheurs à étudier les interactions entre humain et robot afin que les robots humanoïdes du futur puissent présenter des capacités sociales [Nis07].

3.4.1 Principes généraux

La problématique de la qualité de l'interaction sensorimotrice entre humain et robot a beaucoup de points communs avec celle des interactions sensorimotrices entre humain et agent virtuel car les robots et les agents virtuels ont beaucoup de ressemblances. Les applications de réalité virtuelle tendant à être de plus en plus réaliste, les environnements dans lesquels évoluent les agents virtuels interactifs sont de plus en plus similaires aux environnements réels auxquels sont confrontés les robots (environnement en trois dimensions soumit à la gravité, notion de collision, etc.). La réponse en temps réel est cruciale pour chacun d'eux. Enfin, le robot humanoïde et l'agent virtuel expressif sont tous deux créés à l'image de l'homme et possèdent donc les mêmes contraintes morphologiques. Les robots pouvant se résumer à des agents capables d'interactions sensorimotrices, les architectures cognitives développées pour eux ont donc de grandes chances de correspondre aux critères de choix évoqué précédemment. Les architectures robotiques ne forment pas une catégorie d'architecture précise mais étant donné que l'approche symbolique classique est sévèrement critiquée pour son détachement du monde réel, elles se classent généralement soit dans la catégorie émergentiste soit dans la catégorie hybride. Une notion essentielle de ces architectures est la notion d'*embodiment*. Selon les principes de l'*embodiment* notre position corporelle influence notre façon de percevoir et nos expériences sensorielles influencent nos mouvements. Les architectures émergentistes orientées robotique les plus connues sont AAR et EICA, et celles hybrides sont Kismet, HUMANOID, Cerebus et IM-CLEVER. J'ai éliminé l'architecture AAR car elle est jugée par D.Vernon[dGSC10] très complexe et limitée. HUMANOID et Cerebrus sont trop proches des archi-

tectures symboliques et demandent donc trop de connaissances pré-requises pour notre problème. IM-CLEVER fonctionne par renforcement ce qui est inapplicable dans notre cas. L'architecture Kismet est trop focalisée sur l'expression des émotions par le visage (elle possède 21 degrés de liberté pour contrôler la tête). Nous allons maintenant voir les détails de l'architecture EICA.

3.4.2 Exemple d'architecture : EICA

EICA (Embodied Interactive Control Architecture) est basée sur le paradigme *Intention through Interaction* [Nis07]. Selon ce paradigme les intentions doivent être modélisées par des fonctions évoluant de façon dynamique. Et lorsque deux agents interagissent leurs fonctions d'intention co-évoluent

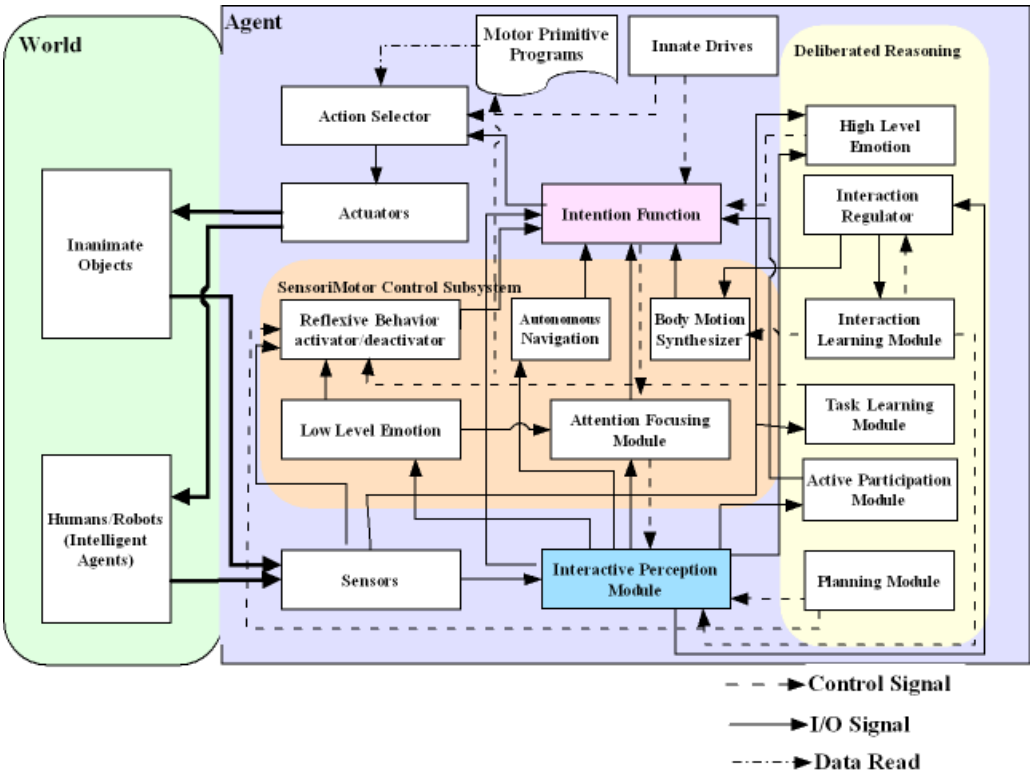

FIGURE 5 – Embodied Interactive Control Architecture (EICA).

créant une intention mutuelle. L'architecture est composée de deux blocs principaux, le sous-système de contrôle sensorimoteur et le sous-système de raisonnement délibératif respectivement en orange et jaune sur la Figure 3. Le premier bloc s'occupe du traitement à court terme des signaux provenant

des capteurs, le second s'occupe du long terme. Cette architecture sépare les signaux d'entrée en deux groupes, d'un côté ceux provenant des agents intelligents et de l'autre ceux provenant des objets inanimés. Cette séparation permet de traiter les agents intelligents différemment, le signal passe par le module de perception interactive, permettant la création d'une intention mutuelle.
Cette architecture peut par exemple être utilisé pour créer un robot-aspirateur contrôlé par des gestes naturels. Le robot analyse les données acquises d'une caméra et actionne ses moteurs pour se déplacer en conséquence. L'article n'explique pas plus en détails cette application.

3.5 Choix de l'architecture

Une des finalités de ce chapitre est de sélectionner l'architecture cognitive la plus à même de résoudre la problématique de ce stage, une interaction crédible entre humain et agent virtuel. Avant d'effectuer ce choix il est nécessaire de rappeler les différentes caractéristiques essentielles établies dans jusqu'à présent ainsi que le contexte du stage. Le couplage sensorimoteur doit être en temps réel, nous ne connaissons que très mal la façon dont les interactions évoluent (nécessité de l'énaction), l'équilibre entre autonomie et synchronie doit être garanti (pour la crédibilité).
Les architectures émergentistes et hybrides remplissent une bonne partie de ces critères, particulièrement celles orientées robotique. Mais les architectures hybrides, en intégrant des notions symboliques, perdent un peu de leur autonomie (elles nécessitent quelques connaissances initiales). Comparé à la nature basique de l'interaction gestuelle, les architectures hybrides peuvent paraître trop complète pour ce problème (prise en compte des émotions, raisonnement logique, planification de suite de tâches dans le but d'accomplir des objectifs, etc.). D'autant plus que nous traiterons l'interaction de façon minimaliste. C'est pour ces raisons que je vais orienter mes travaux sur l'étude d'une architecture plus basique de type émergentiste. Les réseaux de neurones récurrents à temps continu que j'ai décris plus haut respectent parfaitement le critère temps réel, c'est même leur mode normal de fonctionnement (temps continu). Ils n'ont besoin d'aucune connaissance préalable pour fonctionner et grâce à la constante de temps τ spécifique à chaque neurones, ils ont le potentiel pour adopter des comportements parfois synchrones parfois autonome. Ils araîssent donc être les mieux adaptés au contexte de couplage dynamique corporel.

4 Évolution de la configuration du CTRNN

Le fonctionnement de CTRNN composés de plus de deux neurones et qui reçoivent des entrées variables dans le temps échappe aux analyses courantes ce qui rend impossible un paramétrage à la main [Man11]. Pour paramétrer au mieux le réseau de neurones nous utilisons une approche évolutionnaire : les algorithmes génétiques.

S'inspirant de l'évolution naturelle, ces algorithmes font évoluer une population d'individus sur de nombreuse générations en faisant se reproduire les individus adaptés à leur environnement et en faisant mourir les autres celons les principes de la sélection naturelle. Dans notre cas, un individu représentera un jeu de paramètres pour notre CTRNN : les valeurs des différentes inconnues de l'équation différentielle qui définie le comportement des CTRNN (équation 1). Cette solution nous permet de faire évoluer notre réseau de neurones jusqu'à ce qu'il respecte le mieux les critères de notre choix, critères que je détaille dans la suite de cette partie. L'ensemble de ces critères définit la fonction de fitness qui permet d'évaluer et de classer les différents paramétrages de notre réseau en fonction de leur adaptation au problème que nous étudions. On crée donc une population d'individus, initialement aléatoire, dans laquelle on va sélectionner les meilleurs. On utilise une sélection par tournois de deux individus proche géographiquement. La procédure consiste à choisir un individu A aléatoirement parmi la population puis de choisir un autre individu B proche de A. On évalue ensuite A et B grâce à la fonction de fitness. Le vainqueur du tournoi est celui qui obtient le score le plus élevé et il est conservé pour la génération suivante. Le perdant est quant à lui croisé avec le vainqueur, avec une probabilité au alentours de 50% dans notre cas, puis il subit des mutations, à hauteur d'environ 1%. Avec cette technique il n'y a pas besoin de trier toute la population et elle permet également de paralléliser facilement l'évaluation des individus. C'est en simulant cette population sur de très nombreuses générations que le processus d'évolution va maximiser la fitness des individus. En prenant le meilleur individu de la population finale on obtient une configuration, un jeu de paramètres, correspondant à un réseau de neurones ayant un comportement adapté au problème étudié.
Le plus gros travail est de bien définir la fonction de fitness. Elle prend en entrée un individu et retourne sa note d'adaptation, plus celle-ci est élevée plus l'individu est adapté à son environnement. C'est donc par les critères définis dans cette fonction que la sélection naturelle va s'opérer. Durant la fonction de fitness de notre algorithme génétique nous utilisons des enregistrements de trajectoires de souris contrôlée par un humain avec lesquels le

réseau en cours d'évaluation va interagir. Ces enregistrements ont été fait grâce à la même application qui nous permet de tester nos différents algorithmes (application avec les disques contrôlées par la souris, voir partie 2). Ils comprennent la position sur l'axe des abscisses et sur l'axe des ordonnées, nommée X et Y, des deux humains qui interagissent, ainsi que leur vitesse sur ces deux axes. Notre fonction de fitness va donc déterminer la trajectoire du CTRNN en lisant point par point un enregistrement. Pour chacun de ces points elle va fixer les valeurs d'entrée du réseau grâce aux données de l'enregistrement et actualiser les sorties de celui-ci en utilisant l'équation différentielle (1). Ces sorties nous permettent d'obtenir la nouvelle position du CTRNN. Sur chacun de ces pas de simulation la fonction de fitness va attribuer des bonus ou des malus à la note du réseau étudié en fonction de la qualité du déplacement en cours.
Comme on peut le voir dans les équations (1) et (2) les CTRNN sont caractérisés par les poids des différentes liaisons entre les neurones (n^2 poids pour n neurones), leur constante de temps (τ) et leur biais (b). Le nombre de neurones qui composent le CTRNN que nous avons utilisé pour notre application a été fixé à 12. Il a évolué en fonction de la complexité du comportement qu'on lui imposait au travers de la fonction de fitness. Dans notre algorithme génétique, chaque solution est donc représentée par un génome composé de $n^2 + n + n$ gènes ($w + \tau + b$). Ce qui correspond, pour 12 neurones, à 168 paramètres à faire varier pour chaque individu de la population de notre algorithme génétique qui en compte une trentaine. Compte tenu du grand nombre de variables, il a été essentiel d'optimiser le temps de calcul et donc de pré-calculer les fonctions gourmandes en temps comme la fonction de sigmoïde (équation 2) afin d'obtenir des résultats dans un temps raisonnable.

4.1 Les pièges de l'approche évolutionnaire

Les algorithmes génétiques mettent parfois plusieurs jours avant de converger, point où l'on n'arrive plus à améliorer la note de fitness. Dans notre application il faut au moins simuler jusqu'à la 300 000 à 500 000^{ieme} génération avant d'atteindre ce point, cela représente deux à quatre jours de calculs. Il est alors important de ne pas faire d'erreur dans le code sous peine de perdre ces quelques jours. Mise à part les erreurs classiques de codage (fautes de frappes, etc.), il y a d'autres pièges à éviter dus à l'utilisation d'algorithmes génétiques pour paramétrer nos réseaux de neurones. En effet, en utilisant ces algorithmes on se rend compte que les réseaux trouvent toujours la solution la plus simple pour arriver à bout du problème posé, et cette solution n'est pas forcément celle à laquelle on avait pensé au départ. Par exemple, une des premières versions du CTRNN consistait à prendre en entrée du réseau de

neurones la position du premier humain de l'enregistrement et à récompenser le réseau de neurones si sa position était proche de celle du deuxième humain de l'enregistrement. Le réseau comportait donc deux entrées, la position en X et en Y de l'humain A, et deux sorties, la nouvelle position du CTRNN en X et en Y. La fonction de fitness attribuant une note plus ou moins élevée aux réseaux en fonction de leur distance par rapport à l'humain B.

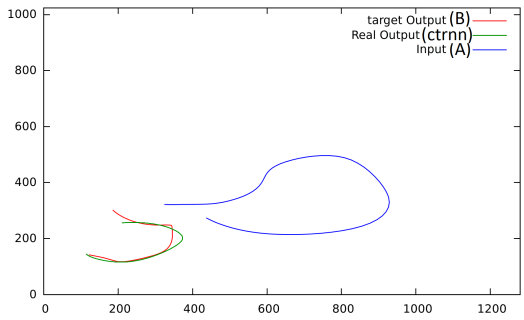

FIGURE 6 – Apprentissage sur un seul enregistrement.

Comme on peut le voir sur la figure (6) le réseau de neurones arrive correctement à reproduire la trajectoire de l'humain B. Mais contrairement à ce que l'on pourrait croire, il ne prend pas en compte la position de l'autre humain qui lui est pourtant fournie en entrée. Si l'on teste notre réseau sur d'autres enregistrements on s'aperçoit (figure 7) qu'au lieu de trouver une relation entre les déplacements de l'humain A et ceux de l'humain B, il a simplement appris la forme de la trajectoire sur laquelle il s'est entraînée et la répète quelque soit l'entrée qu'on lui donne. Il est donc nécessaire d'exercer les réseaux de neurones sur différents échantillons si l'on veux trouver une solution générale à notre problème. Dans un réseau de neurones classique on utiliserait par exemple un jeu d'un millier de valeurs pour s'entraîner, mais étant donné la nature des CTRNN (**Continuous-Time** Recurrent Neural Network) ceux-ci ne prennent en entrée non pas une valeur mais une suite de valeurs et c'est donc un millier d'enregistrements qu'il faudrait utiliser pour obtenir une solution parfaitement généraliste. Compte tenu de la difficulté d'enregistrer autant d'interactions humain avec humain et surtout des temps de calculs énormes qu'il serait nécessaire d'effectuer pour chaque fitness, les enregistrements que nous utiliserons par la suite représenteront environ une quinzaine de secondes pour 800 pas de simulation. C'est pourquoi nous avons choisi d'utiliser seulement trois enregistrements différents par calcul de fitness.

Un autre piège qui survient fréquemment est la mauvaise définition de notre problèmatique au travers des critères décrits dans la fonction de fitness.

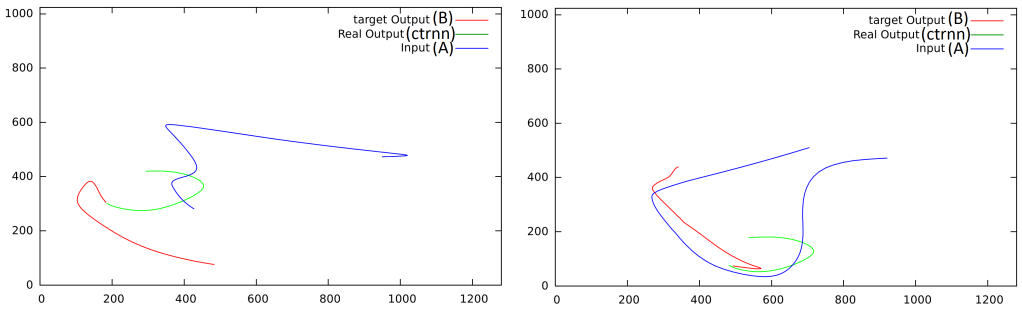

FIGURE 7 – Apprentissage de la forme.

Par exemple, si l'on veut que le CTRNN se déplace aléatoirement sur l'écran, en prenant quand même en entrée les déplacements d'un humain, on va lui demander d'avancer en permanence et qu'il n'aille pas tout droit. Ce qui se traduit, dans la fonction de fitness, par une pénalité sur la note du réseau si celui-ci a une vitesse nulle ou un angle de rotation nul. Plutôt que d'errer aléatoirement sur l'écran le réseau va se fixer une vitesse en X non nulle, une vitesse en Y non nulle et un angle de rotation non nul. Il va alors suivre une trajectoire elliptique à l'infini (figure 8). Il ne faut donc pas seulement lui dire quoi faire mais également lui dire ce qu'il ne doit pas faire. Dans ce cas, il ne faut pas oublier de pénaliser le CTRNN si sa vitesse et son angle de rotation ne varie pas d'un pas de simulation à l'autre. Mais là encore il peut se mettre à osciller en allant un coup à gauche, un coup à droite, etc. Il faut alors vérifier sur plusieurs pas en arrière qu'il ne se répète pas.

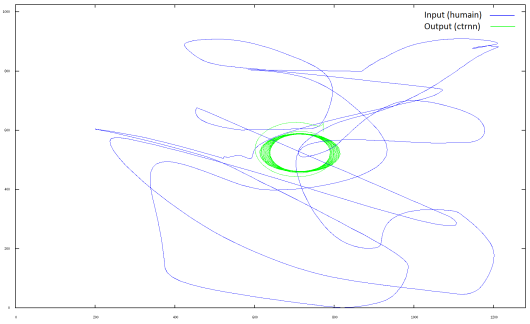

FIGURE 8 – Fonction de fitness trop simple.

4.2 Réalisme des mouvements

Si l'on analyse les résultats du stage précédent, on s'aperçoit que tous les algorithmes utilisés jusque là se font détecter, non pas par une interaction

incohérente avec l'humain, mais plutôt par des trajectoires irréalistes. Les utilisateurs de l'application ont rapidement remarqué que les algorithmes avaient soit des temps de réaction quasiment nuls, soit des mouvements très répétitifs, soit des trajectoires parfaitement rectilignes d'une précision irréalisable par un humain. Il m'a donc parut important de travailler sur ces points avant même de commencer à faire interagir le CTRNN avec des humains.
En analysant plusieurs tracés humain contre humain, j'ai remarqué que l'on pouvait découper les trajectoires en différents arcs de cercles et c'est pour intégrer ce fonctionnement au réseau de neurones que j'ai décidé de contrôler celui-ci en vitesse et en rotation. Le réseau prend en entrée la position en X et en Y de l'humain et on lit en sortie du réseau sa vitesse en X et en Y et sa vitesse de rotation. Le CTRNN possède ici deux entrées et trois sorties. La figure 9 est un exemple de trajectoire d'un réseau contrôlé de cette façon.

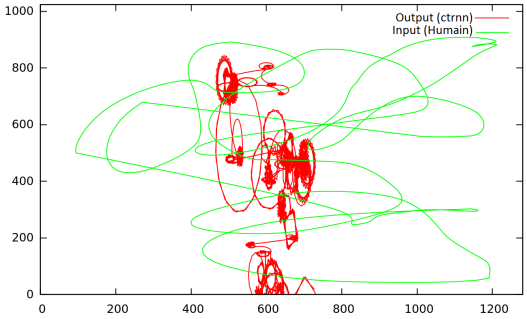

FIGURE 9 – Trajectoire contrôlée en vitesse et en rotation.

Sur cette figure on remarque que l'algorithme est mal paramétré puisque les déplacements du réseau de neurones sont beaucoup trop rapide. On commence donc par limiter la vitesse suivant l'axe X et Y ainsi que la vitesse de rotation en pénalisant le CTRNN dès que ses vitesses franchissent un seuil. Les seuils sont fixés de manière arbitraire, déterminés en étudiant les vitesses de la souris contrôlée par un humain sur les mêmes enregistrements que ceux utilisés par l'algorithme génétique. La solution de facilité de l'algorithme génétique pour ne pas faire d'excès de vitesse est d'avoir une vitesse nulle. Donc, en plus de limiter la vitesse, on pénalise aussi les vitesses insuffisantes ainsi que les temps d'arrêts prolongés. Pour éviter les problèmes vus sur la figure 8 on pénalise aussi les vitesses et rotations invariantes. On rajoute également une contrainte de direction pour l'obliger à changer son sens de rotation de temps en temps. Il ne manque maintenant plus qu'à imposer les limites de l'écran au réseau de neurones, puisque dans l'application le disque qui représente la souris ne peut pas sortir de l'écran. Si la trajectoire sort du

cadre le disque va se planter dans un coin de la fenêtre. On pénalise donc le réseau dè qu'il sort des limites de l'écran et pour éviter qu'il ne se recroqueville au milieu de l'écran on lui donne la consigne inverse : plus il s'étend (dans les limites de l'écran) plus on le récompense.

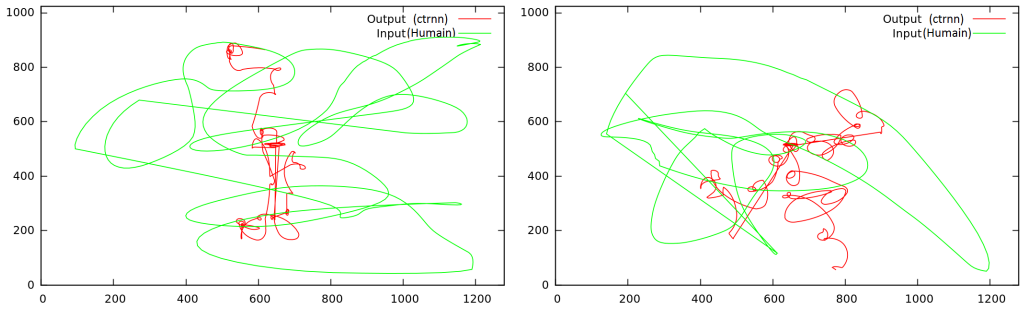

FIGURE 10 – Comportement respectant des contraintes de vitesse et de rotation.

Après l'application de toutes ces pressions de sélection, on obtient les trajectoires de la figure 10. On constate que le CTRNN se déploie bien sur l'espace qui lui est attribué, il se rapproche des bords sans les dépasser, et que ses trajectoires ne sont pas totalement extravagantes. Mais bien qu'il respecte les limites de vitesse qu'on lui a imposé, il ne suit pas encore certaines lois liées au mouvements humain comme la loi de Viviani.

4.3 Prise en compte de la loi de Viviani

Dans un de ses articles [PV91] P.Viviani étudie la géométrie du mouvement de dessin. Lors d'expérimentations il demande à des enfants et des adultes de dessiner de façon continue des ellipses de différentes tailles afin d'étudier la relation entre la vitesse de tracé et le rayon de courbure de la trajectoire. Il montre que dès l'age de cinq ans le mouvement de la main d'un humain suit une loi mathématique dite *two-thirds power law*, loi de puissance deux tiers (équation 3).

$$V(t) = KR(t)^{\beta}, \qquad \beta \geq 0 \tag{3}$$

Avec K le facteur de gain de vitesse, V(t) la vitesse et R(t) le rayon de courbure. Dans son article Viviani détermine que $\beta = \frac{2}{3}$ est une bonne valeur pour estimer les mouvements humain, c'est cette valeur que nous retiendrons pour notre algorithme. 0D'après la cinématique nous obtenons les équations (4) pour la vitesse et (5) pour le rayon de courbure.

$$V(t) = \sqrt{(\frac{\delta x}{\delta t})^2 + (\frac{\delta y}{\delta t})^2} \tag{4}$$

$$R(t) = \frac{V^3}{(\frac{\delta^2 y}{\delta t^2})(\frac{\delta x}{\delta t}) - (\frac{\delta^2 x}{\delta t^2})(\frac{\delta y}{\delta t})} \tag{5}$$

$\frac{\delta x}{\delta t}$, $\frac{\delta y}{\delta t}$, $\frac{\delta^2 x}{\delta t^2}$ et $\frac{\delta^2 y}{\delta t^2}$ étant respectivement la vitesse et l'accélération suivant l'axe X et Y.

Dans la fonction de fitness de notre algorithme génétique nous connaissons la vitesse en X et en Y du réseau et en la comparant à la différence de vitesse d'un pas de simulation à l'autre on en déduit l'accélération sur ces deux axes. Nous avons donc en notre possession toute les valeurs pour le calcul du K de l'équation (3). Il ne reste plus qu'à analyser les différentes valeurs de K sur le tracé enregistré d'un humain utilisant sa souris pour déterminer des valeurs caractéristiques qui permettrons d'évaluer nos réseaux de neurones.

Pour se faire, nous utilisons *gnuplot* dans le but de dessiner une courbe en trois dimensions de l'évolution de K en fonction du parcours d'un humain (figure 11). L'image du haut est la vue de dessus de cette courbe, elle représente la trajectoire de l'humain suivant l'axe X et Y. L'axe Z représente quant à lui la valeur de K. Sur l'image du bas, vue de côté, on peut voir que cette valeur subit des pics lorsque la trajectoire tourne trop violemment. C'est ces pics que nous souhaitons éviter pour le CTRNN. Dans la fonction de fitness, nous résolvons donc l'équation (3) pour chaque pas de simulation et appliquons une pénalité au réseau s'il dépasse le seuil en orange dans la figure (11). Sur cette figure la valeur de K atteint des pics plus élevé qu'elle ne devrait, puisque c'est une courbe réalisé par un humain, cela est du aux erreurs de capture de la souris. En enregistrant seulement une cinquantaine de points par seconde, le calcul de la dérivée de la vitesse est faussé lorsque la souris change de direction vraiment très rapidement.

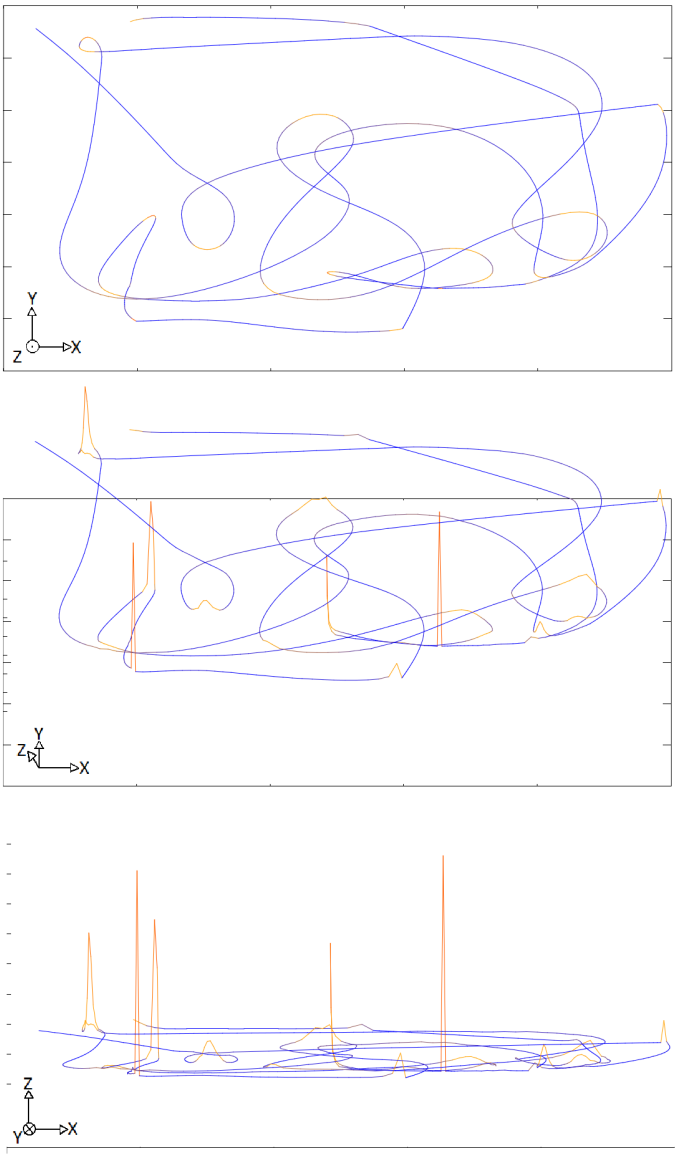

FIGURE 11 – Courbe 3D de la loi de Viviani sur un tracé humain, vue sous trois angles différents.

Après la mise en œuvre de ce critère de pression dans la fonction de fitness, en plus des critères pour le réalisme des mouvements, on obtient la trajectoire de la figure (12). Dans l'ensemble la trajectoire parait correcte mais il manque toujours un point essentiel : le réseau ne prend pas encore en compte de comportement de l'humain, il n'y a donc pas d'interaction.

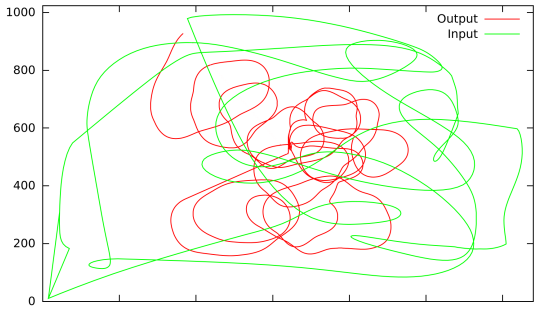

FIGURE 12 – CTRNN avec application de la loi de viviani.

4.4 Début d'interaction : suivi de trajectoire

La première étape avant la mise en place d'une interaction complexe avec un humain est de prouver que le CTRNN est capable de prendre en compte la trajectoire de cet humain. Pour cela nous avons créé un réseau de neurones dont l'unique objectif est de se rapprocher de la position de l'humain, il va donc adopter un comportement de suiveur. Jusqu'à présent nous fournissions en entrée du réseau la position de l'humain. Mais étant donné que le réseau ne connaît pas sa propre position, il lui est impossible de se déplacer vers l'humain dans cette configuration. Pour palier ce problème nous n'allons plus donner la position mais la distance séparant le CTRNN de l'humain. Les deux nouvelles entrées sont donc $(xHumain - xCTRNN)$ et $(yHumain - yCTRNN)$. Après plusieurs tests nous nous sommes aperçus que le réseau avait du mal à suivre l'humain en étant contrôlé en vitesse et en rotation. Nous avons donc décidé de modifier une nouvelle fois la structure du réseau de neurones en contrôlant les déplacements seulement par la vitesse en X et Y, réduisant le nombre de sorties à deux.

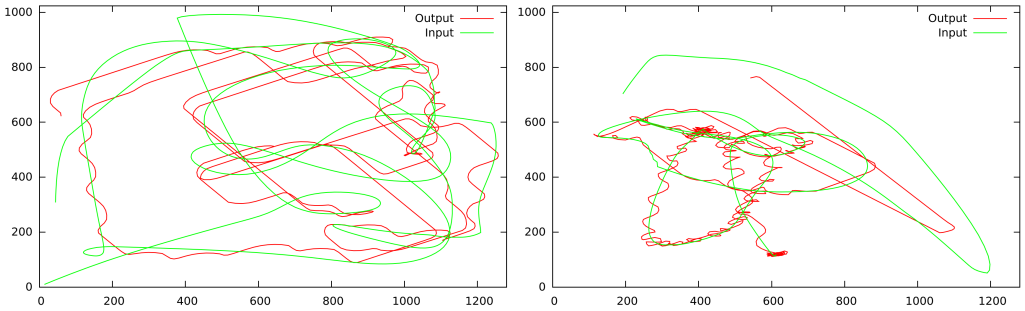

FIGURE 13 – Trajectoire sous contrainte de distance avec l'humain.

Les déplacements du réseau suivent maintenant clairement la trajectoire de l'humain, comme on peut le voir sur la figure (13). Le critère de suivis

est formulé très simplement dans la fonction de fitness puisqu'il suffit de pénaliser le réseau en fonction de la distance qui le sépare de l'humain.

4.5 Corrélation des trajectoires : Cross Correlation

L'hypothèse principale de mon stage est qu'il existe un couplage lors de l'interaction entre deux humains. Suivant cette hypothèse on peut conjecturer qu'il existe une corrélation entre la trajectoire du premier humain et celle du deuxième. Pour attester cela nous avons donc appliqué des formules de corrélations croisées aux trajectoires de deux humains interagissant. Équation de la cross correlation en fonction du retard d ($delay$) :

$$r(d) = \frac{\sum_i [(x(i) - mx) * (y(i-d) - my)]}{\sqrt{\sum_i (x(i) - mx)^2} \sqrt{\sum_i (y(i-d) - my)^2}} \tag{6}$$

Avec $x(i)$ et $y(i)$ les deux séries de points à comparer, mx et my respectivement les moyennes des séries $x(i)$ et $y(i)$ et d le retard appliqué à la série $y(i)$.
La corrélation croisée mesure la similarité entre deux séries de points en fonction du retard appliqué à l'une de ces deux séries. Par exemple, sur la figure (14) la valeur de la cross correlation entre la courbe verte et la courbe bleu est maximale lorsque le retard vaut trois car si l'on décale la courbe verte de trois unités de temps, les deux courbes se superposent parfaitement. La valeur en $d = 3$ est de 1 car cette fonction est normalisée.

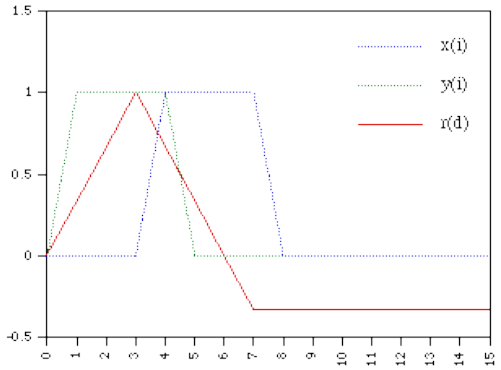

FIGURE 14 – Corrélation en position.

En retranscrivant l'équation (6) sous *scilab* et en important les enregistrements d'interaction humain avec humain, on est maintenant en mesure de

visualiser la corrélation des trajectoires.

La figure (15) représente les variations des positions en X, vert clair pour l'humain A et vert foncé pour l'humain B, et en Y, rouge pour l'humain A et orange pour l'humain B. La corrélation croisée des positions en X et celle des positions en Y sont affichées simultanément sur la figure (16) respectivement en vert et rouge.

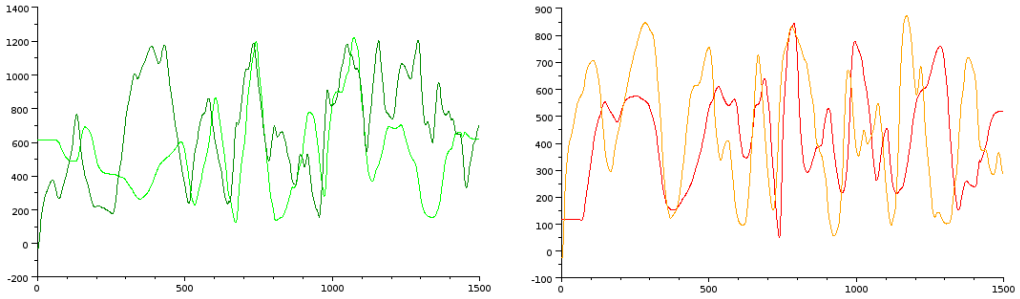

FIGURE 15 – Position en X (vert) et Y (rouge).

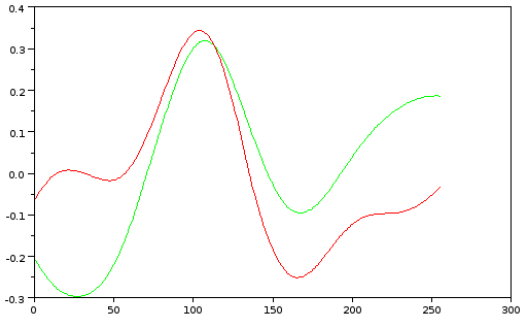

FIGURE 16 – Corrélation en position en X et Y.

Il existe bien une corrélation entre les positions des deux humains car si elles n'étaient pas corrélées, la valeur de la corrélation croisée serait inférieure à zéro. La valeur maximale est atteinte à un délai d'environ cent pas de simulation, en étudiant ce type de corrélation sur une dizaine d'autres enregistrements, j'établis que ce pic se situe très fréquemment entre vingt et cent-vingt pas de simulation. De la même manière je constate que la hauteur du pic dépasse à chaque fois la valeur de $0,25$.
Étant donné que le calcul de la corrélation croisée est relativement chronophage, il n'est pas raisonnable de le calculer sur l'ensemble de la plage de délai $[20; 120]$. On se contentera donc d'échantillonner cette plage aux valeurs $(20; 40; 80; 120)$, la largeur des pics étant suffisamment importante pour se le

permettre. Dans chaque fonction de fitness nous calculerons donc quatre valeurs de corrélation et nous augmenterons la note des réseaux qui dépassent le seuil de $0,25$.
Une forte corrélation en position se manifeste dans l'application par un suivi de la trajectoire de l'autre. Une corrélation simultanée en X et en Y signifie un suivi exact de la trajectoire, alors qu'une corrélation sur un seul axe peut signifier une recopie à distance des mouvements.

Nous suivons le même procédé pour analyser la corrélation en vitesse. La figure (17) représente les variations des vitesses en X, vert clair pour l'humain A et vert foncé pour l'humain B, et en Y, rouge pour l'humain A et orange pour l'humain B. La corrélation croisée des vitesses en X et celle des vitesses en Y sont affichées simultanément sur la figure (18) respectivement en vert et rouge.
On examine de nouveau ces courbes avec une dizaine d'enregistrements différents et on constate, logiquement, que la plage de délai est la même que pour la position, ainsi que la valeur des pics.

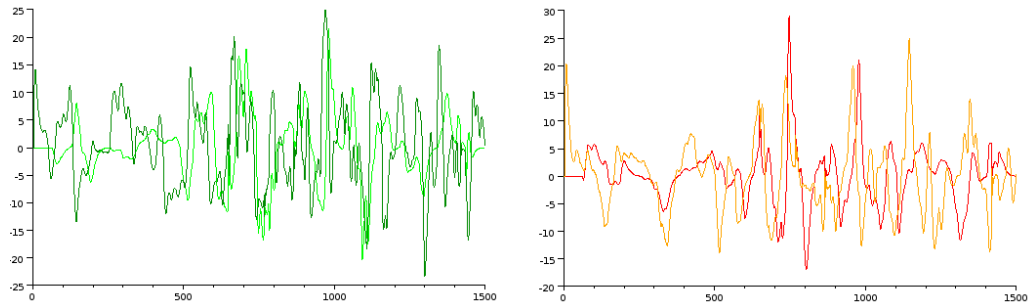

FIGURE 17 – Vitesse en X (vert) et Y (rouge).

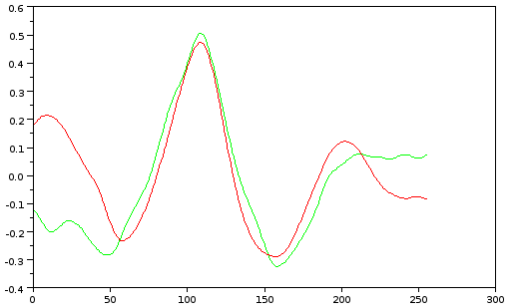

FIGURE 18 – Corrélation en vitesse.

Une corrélation en vitesse se manifeste dans l'application par une recopie des mouvements de l'autre, sans forcément être au même endroit sur l'écran.

4.6 Version finale de la fonction d'évaluation

En rassemblant tous les critères susdit au sein d'une même fonction d'évaluation on obtient une fitness assez complexe qu'il est nécessaire de paramétrer. En effet, il faut pondérer correctement les pénalités et les bonus donnés à la note de fitness sous peine de voir un critère écraser tous les autres.
La figure (19) montre deux exemples de trajectoires de CTRNN qui ont évolué sous contrainte de tous ces critères. Cette version du CTRNN n'est pas la version finale, mais c'est la version qui était la plus avancée à la date des expérimentations et c'est donc celle qui a été testée. Le côté "suiveur" du réseau est trop prononcé, ce qui le rend "collant".

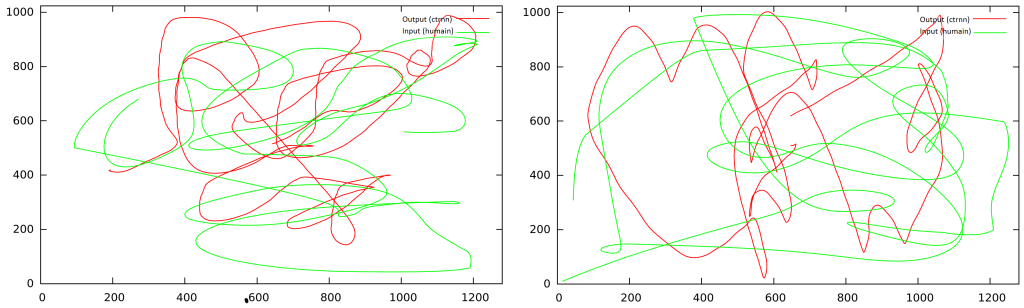

FIGURE 19 – Trajectoire finale du réseau de neurone.

4.7 Enregistrement des réseaux

Toutes les configurations finales de réseaux générées par l'algorithme génétique sont sauvegardées dans un format lisible par le logiciel *Graphviz*. Ce logiciel permet de dessiner des graphiques orientés comme les réseaux de neurones. La figure (20) est un exemple de représentation d'un CTRNN comportant douze neurones, deux entrées (en haut) et trois sorties (en bas). L'épaisseur des arcs représente le poids des différentes liaisons, la taille de la lettre "B" sur les nœuds représente la valeur du biais et leurs teinte de rouge indique la valeur du τ.

Aucune de ces structures de réseau n'a pour l'instant été analysée, pour des raisons de temps et surtout parce que ces structures évoluent sans arrêt depuis le début du stage. Le fichier que crée l'algorithme génétique étant formaté, il est très facile de récupérer les données de configuration du réseau de

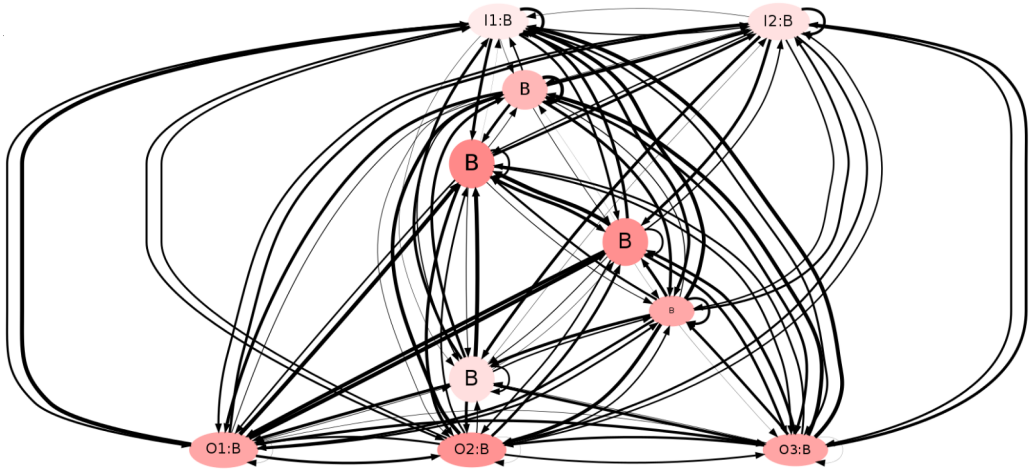

FIGURE 20 – Structure du réseau de neurones.

```
digraph G{
        node [style=filled];
        0 -> 0 [penwidth=3.594038];
        0 -> 1 [penwidth=1.547069];
        0 -> 2 [penwidth=0.506915];
        0 -> 3 [penwidth=3.463297];
        0 -> 4 [penwidth=0.487165];
        0 -> 5 [penwidth=3.644756];
        0 -> 6 [penwidth=2.828123];
        0 -> 7 [penwidth=2.538648];
        0 -> 8 [penwidth=2.410091];
        0 -> 9 [penwidth=2.499114];
        1 -> 0 [penwidth=0.532020];
        1 -> 1 [penwidth=3.164659];
        1 -> 2 [penwidth=0.571875];
        1 -> 3 [penwidth=1.754532];
        1 -> 4 [penwidth=3.573910];
        1 -> 5 [penwidth=1.897595];
        1 -> 6 [penwidth=3.512143];
        1 -> 7 [penwidth=2.498799];
        1 -> 8 [penwidth=2.534265];
        1 -> 9 [penwidth=2.504279];
        2 -> 0 [penwidth=1.863087];
        2 -> 1 [penwidth=4.378920];
        2 -> 2 [penwidth=4.094495];
        2 -> 3 [penwidth=3.278285];
        2 -> 4 [penwidth=1.439893];
        2 -> 5 [penwidth=3.508834];
        2 -> 6 [penwidth=0.829259];
        2 -> 7 [penwidth=3.741705];
        2 -> 8 [penwidth=0.733035];
        2 -> 9 [penwidth=3.203564];
        3 -> 0 [penwidth=0.124273];
        3 -> 1 [penwidth=3.198852];
        3 -> 2 [penwidth=0.880715];
        3 -> 3 [penwidth=2.469876];
        3 -> 4 [penwidth=0.944246];
        ....
        ....
        0 [color="0.0 0.081318 1.0", label="B", fontsize="17.626850"];
        1 [color="0.0 0.120771 1.0", label="B", fontsize="18.906565"];
        2 [color="0.0 0.281880 1.0", label="B", fontsize="20.366942"];
        3 [color="0.0 0.464414 1.0", label="B", fontsize="26.736972"];
        4 [color="0.0 0.125072 1.0", label="B", fontsize="25.554156"];
        5 [color="0.0 0.343028 1.0", label="B", fontsize="9.806638"];
        6 [color="0.0 0.434780 1.0", label="B", fontsize="25.867947"];
        7 [color="0.0 0.350463 1.0", label="B", fontsize="17.087027"];
        8 [color="0.0 0.424509 1.0", label="B", fontsize="16.951442"];
        9 [color="0.0 0.352515 1.0", label="B", fontsize="15.845577"];
        {rank = min; 0; 1;}
        {rank = max; 7; 8; 9}
        0 [label="I1:B"];
        1 [label="I2:B"];
        7 [label="O1:B"];
        8 [label="O2:B"];
        9 [label="O3:B"];
}
```

FIGURE 21 – Fichier formaté pour graphviz.

neurones et de les utiliser dans un autre logiciel comme celui des expérimentations.

5 Les expérimentations

Pour évaluer la crédibilité de notre algorithme utilisant les CTRNN, nous avons procédé à des expérimentations. Elles ont permis de situer les performances de celui-ci par rapport à d'autres algorithmes et sur différents critères. L'interface utilisé est la même que lors du stage précédent avec quelques modifications. Les expérimentations ont eu lieu à l'ENIB (Ecole Nationale d'Ingénieur de Brest), tous les sujets étant étudiant de cette école ils ont tous environ vingt ans et utilisent tous régulièrement des ordinateurs. Nous avons néanmoins enregistré leur age et leur niveau de familiarité avec les jeux multi-joueurs en ligne (familier des interactions avec des avatars) lors d'un questionnaire pré-expérimentation.
Chaque expérimentation est composée de six phases d'interactions de quarante secondes dont une seule avec un humain, durant les cinq autres phases d'interactions les sujets font face à différents algorithmes. Les sujets ne savent pas contre combien d'humains ils vont interagir. On étudie les interactions entre deux humains, il est donc nécessaire de faire passer tous les sujets deux par deux. L'application se divise alors en deux parties, serveur et client. C'est le serveur qui décide de l'ordre dans lequel les deux sujets vont se confronter aux différents algorithmes. Tout d'abord il choisi aléatoirement le tour durant lequel les deux sujets seront l'un contre l'autre, puis il fixe, toujours aléatoirement, l'ordre de passage des autres algorithmes qui ne sont alors pas forcément les mêmes chez le client et le serveur.

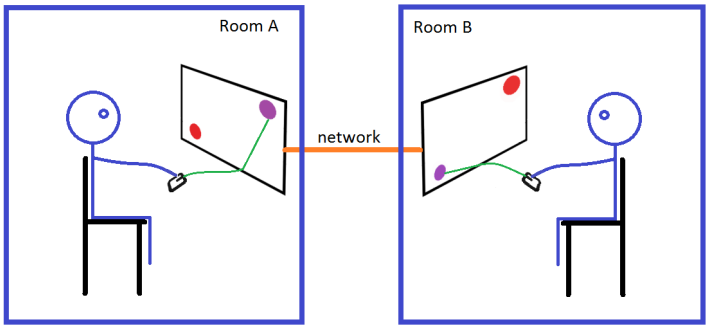

FIGURE 22 – Organisation de l'expérimentation.

La principale consigne des sujets est d'interagir avec l'autre à l'aide de leur souris dans le but de déterminer si ils ont affaire à un humain ou à une machine. Lors de chaque phase d'interaction il leur est demandé d'appuyer sur la touche "H", comme Humain, ou sur la touche "M", comme Machine, dès qu'il sont sûrs de la nature de leur opposant. Le temps qu'ils mettent à se décider est ainsi enregistré. Il sagit dune version adaptée à notre cas du

BIP (Break In Presence) de Slater [MS00]. Ils leur est également demandé de continuer à interagir après avoir fait leur choix entre humain ou machine pour ne pas perturber l'expérimentation de leur binôme, dans l'éventualité où ils interagiraient ensemble. A la fin de chacune des phases d'interactions un questionnaire numérique est soumis aux sujets. Au tout début de l'expérience une phase de test est proposée aux sujets ce qui leur permet de se familiariser avec l'interface et avec la dynamique d'interaction, l'application est programmée de façon à gérer les collisions pour que les deux curseurs puisse se "pousser" l'un l'autre. Au cours de ce test rien n'est enregistré et on ne pose aucune question aux sujets.

5.1 Les différents algorithmes

Comme on peut le voir dans la partie 4, nous avons essayé d'intégrer deux aspects importants de l'interaction entre humain et avatar, le réalisme des mouvements et la prise en compte de l'autre. Pour évaluer les performances de notre réseau de neurones nous avons choisi de soumettre aux sujets des algorithmes ne présentant chacun qu'une partie de ces aspects.

5.1.1 Miroir

L'algorithme miroir date du stage précédent, il se contente de copier les mouvements de l'humain par symétrie suivant le centre de l'écran. Il est sensé représenter le côté "prise en compte de l'autre" de l'interaction et ce côté seulement car bien que ses mouvements aient une allure humaine, copie de l'autre humain, son temps de réaction instantané et la précision absolue de ses gestes le rendent totalement irréaliste.

5.1.2 Aléatoire

L'algorithme aléatoire ne prend pas en compte la position de l'humain et change de position aléatoirement toute les secondes. Son déplacement possède une certaine inertie pour qu'il ne se téléporte pas d'un point à un autre mais la transition entre deux points reste parfaitement rectiligne. Nous le traitons donc comme un algorithme ne satisfaisant ni le réalisme des mouvements, ni la prise en compte de l'humain.

5.1.3 Enregistrement

Pour montrer le côté réaliste des gestes sans interaction il suffit de lire un enregistrement d'une trajectoire faite par un humain, donc parfaitement réaliste.

Le même enregistrement est utilisé pour tous les sujets. Le sujet faisant face à un enregistrement il n'y a donc pas d'interaction.

5.1.4 Enregistrement sans collision

Comme l'application de l'expérimentation possède un gestionnaire de collision, le sujet peut perturber la trajectoire de l'enregistrement en se mettant sur son chemin. Dès lors on peut considérer qu'il y a une interaction, même faible. C'est pourquoi nous avons décidé de faire une deuxième version de l'algorithme "enregistrement", cette fois en désactivant le gestionnaire de collision. La phase d'interaction avec cet algorithme est donc différente de toutes les autres, y compris celle avec l'humain, puisque c'est la seule sans collision. Mais ces résultats sont à comparer avec ceux de l'enregistrement classique.

5.1.5 CTRNN

Le CTRNN est le réseau de neurones configuré lors de la partie 4, il est sensé prendre en compte le comportement de l'autre humain tout en adoptant un déplacement réaliste.

5.2 Les questions

Le but d'une expérimentation est de récolter des données. Dans notre application nous récoltons des données de cinq sources différentes. Les informations concernant le sujet (son age, son sexe, etc.) sont enregistrées avant le début de l'expérimentation.

À la fin de chaque phase d'interaction un questionnaire est posé aux sujets concernant leurs impressions sur cette dernière. Il est composé de cinq affirmations adaptées du questionnaire de présence sociale de Bailenson [JNBL03] :

- J'ai perçu que j'étais en présence d'une autre personne au travers de l'application.
- J'ai ressenti que le cercle rouge cherchait à me faire réagir.
- L'idée que le cercle mauve N'EST PAS contrôlé par une personne réelle m'a souvent traversé l'esprit.
- Le cercle rouge semble prendre en compte/être sensible à mon comportement.

- J'ai ressenti le comportement du cercle rouge comme étant une trajectoire prédéfinie.

Les sujets doivent donner leur avis sur ces affirmations en utilisant une échelle de Likert de (-3) "Pas du tout d'accord" à (+3) "Tout à fait d'accord" (voir figure 23).

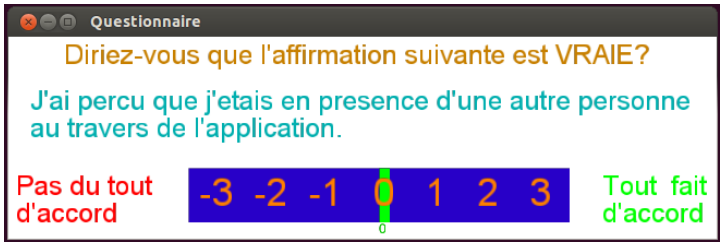

FIGURE 23 – Interface du questionnaire.

L'intégralité des mouvements des deux disques est enregistrée pour des évaluations objectives.

Lors des phases d'interaction on enregistre le temps que mettent les sujets à détecter les humains ou les algorithmes (touche H ou M). On enregistre également leurs changement d'avis.

À la fin de l'expérimentation un questionnaire papier est proposé, il est composé de questions ouvertes :

- Quels sont les critères qui vous permettent de détecter que vous avez affaire à un programme ?
- Quels sont les critères qui vous permettent de détecter que vous avez affaire à une personne réelle ?
- Y a-t-il eu des évènements qui vous ont permis de savoir que vous n'aviez pas affaire à un programme, si oui lesquels ?
- Y a-t-il eu des événements qui vous ont permis de savoir que vous n'aviez pas affaire à une personne réelle, si oui lesquels ?
- Avez-vous des remarques sur l'expérimentation ?

5.3 Résultats

L'expérimentation a été réalisé sur 44 sujets, ce qui a permis d'obtenir des résultats plutôt significatif. Les tests anova sur les données donnent toujours une probabilité de rejet à tord de l'hypothèse nulle de $p - value < 0,05$.

5.3.1 Questionnaire numérique

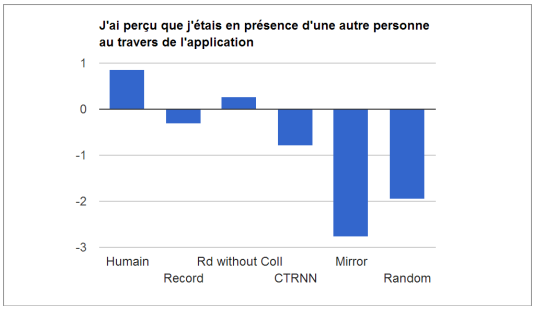

FIGURE 24 – Question 1 : Sentiment de présence.

La figure (24) montre les résultats obtenus par les différents algorithmes à la question sur le sentiment de présence. On constate que c'est l'humain qui remporte le meilleur score (heureusement) et que les enregistrements arrivent tous deux après. Le fait qu'il y est une différence significative entre le score de l'humain et ceux des enregistrements prouve l'importance de l'interaction pour la crédibilité des agents virtuels. Bien que le CTRNN arrive après les enregistrements, il dépasse largement les deux autres algorithmes. L'algorithme miroir étant détecté très rapidement par la grande majorité des sujets, il échoue totalement à ce test. La différence de points entre l'algorithme aléatoire et le miroir s'explique par son côté imprévisible qui le rend plus dur à détecter.

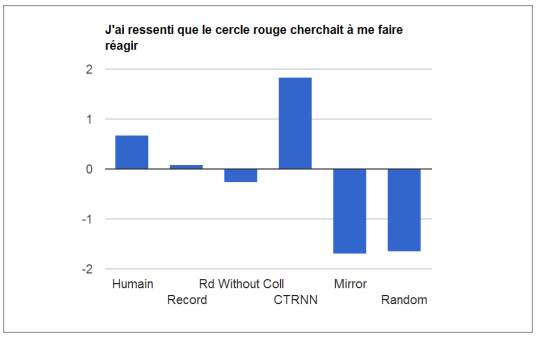

FIGURE 25 – Question 2 : Cherche l'interaction.

Les résultats de la figure (25), qui portent sur le sentiment d'interaction, révèlent que le côté "suiveur" du CTRNN est trop présent. Il passe son temps à chercher l'autre alors qu'un humain passe par des phases où il ne s'occupe pas de l'autre, l'humain obtient d'ailleurs un score deux fois moindre. L'équilibre entre autonomie et synchronie fait partie des notions

importantes retenues dans la partie 2. Les résultats dévoilent aussi que si les sujets ont largement tranchés la question pour les algorithmes miroir et aléatoire, ils restent indécis sur les enregistrements.

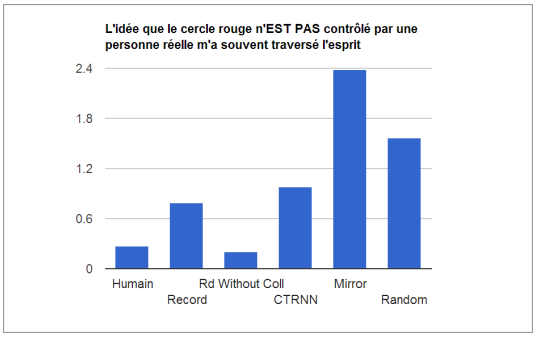

FIGURE 26 – Question 3 : Humain ou machine.

Sur la figure (26), portant sur le sentiment d'interaction avec une machine, on voit de nouveau que l'écart entre le CTRNN et les algorithmes basiques, miroir et aléatoire, est significatif. Les résultats pour l'humain et les enregistrements sont quant à eux clairement meilleurs, mais les différences entre eux ne sont pas éloquentes.

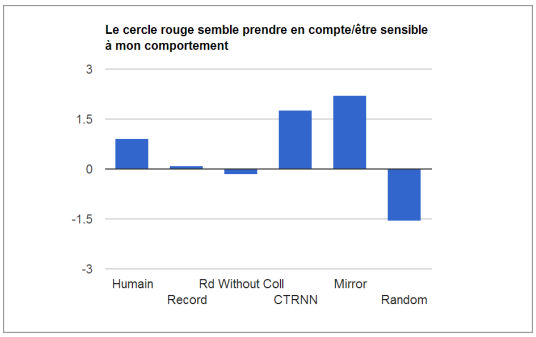

FIGURE 27 – Question 4 : Prise en compte de l'autre.

La figure (27) montre des résultats cohérents quant au sentiment de prise en compte du comportement des sujets. Cette question est une variante de la question 2, la comparaison avec la figure(25) montre des résultats très similaires excepté pour l'algorithme miroir. Cela montre que l'on peut très bien prendre en compte l'autre sans générer de sentiment d'interaction. L'algorithme aléatoire ne tient pas compte de l'humain et il a été classé comme tel. À l'inverse des algorithmes miroir et CTRNN qui ont obtenu une note plus élevée que celle de l'humain dût à leur manque d'autonomie. La seule

exception vient des enregistrements, le fait d'avoir des trajectoires parfaitement réalistes en on trompé plus d'un. Ce qui justifie l'indécision des sujets à cette question.

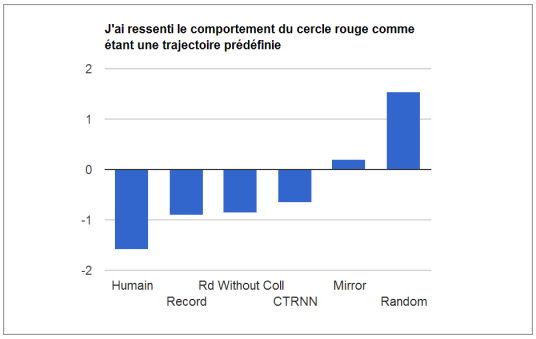

FIGURE 28 – Question 5 : Trajectoire prédéfinie.

En regardant le détail des réponses à la question 5 (figure 28) et d'après des remarques dites à l'oral par certains sujets durant l'expérimentation, on s'aperçoit que cette question n'as pas été comprise de la même façon par tout le monde. Certains ont interprété "Trajectoire prédéfinie" comme une trajectoire enregistrée, une vidéo. D'autre l'on compris comme comportement prédéfini, codé en dur. L'algorithme miroir à été démasqué par tous les sujets, les réponses à la question 5 auraient du être unanimes. Or la moitié des sujets ont répondus l'extrême positif : (+3) et l'autre moitié, l'inverse : (-3). La formulation de cette question devra donc être revue pour les prochaines expérimentations.
Si jamais cela peut être considéré comme un résultat, on constate tout de même que le CTRNN se situe du bon côté, celui de l'humain...

5.3.2 Temps de détection

Ces données n'ont pas encore été traitées au moment de la rédaction de mon rapport, elles le seront d'ici la fin du stage.

5.3.3 Questionnaire post-expérimentation

Ce questionnaire a confirmé ce que l'on savait déjà, les algorithmes sont démasqués par leur comportement répétitif et rectiligne. L'humain répond aux sollicitations et adopte régulièrement un comportement presque aléatoire (équilibre entre autonomie et synchronie).
Quelques remarques proposent d'améliorer l'application en affichant le temps restant pour chaque phase d'interaction car certains ont été surpris par la fin

de ces phases et n'ont pas eu le temps d'appuyer sur les touches H ou M. Plusieurs sujets affirment avoir découvert des algorithmes seulement en s'arrêtant, c'est la plus grosse faiblesse du CTRNN. Les réseaux de neurones étant des systèmes déterministes, si on fige les entrées on fige les sorties. La particularité des CTRNN, dans ce cas là, est de ne pas s'arrêter mais plutôt d'évoluer de façon trop régulière, ce qui est toujours mieux que de se figer. Comme notre CTRNN prend en entrée la distance qui le sépare de l'humain, celle-ci varie même lorsque celui-ci est à l'arrêt. Mais dès lors que le CTRNN repasse à un endroit où il déjà passé, il entre dans une boucle et va répéter cette séquence à l'infini jusqu'à ce que l'humain ce remette à bouger.

6 Conclusion et perspectives

Le but de ce stage était de mettre en place une architecture cognitive capable de reproduire le couplage dynamique temps réel existant lors d'une interaction entre deux humains. L'hypothèse de base était que pour avoir un sentiment de présence de l'autre il fallait qu'il y est à la fois de l'interaction et de l'autonomie. Nous nous sommes contenté d'une interaction minimaliste, ne prenant en compte que les gestes de la main dans un environnement virtuel 2D, afin de bien cerner le problème et de saisir les fondements de ce couplage. Une étude des architectures cognitives existantes a permis de déterminer que le type d'architecture le plus adapté à notre problème est le type émergentiste, et parmi ces architectures ce sont les réseaux de neurones à temps récurrents que nous avons choisi d'implémenter. Étant impossible de paramétrer ces réseaux à la main, nous avons utilisé l'approche évolutionnaire pour obtenir le réseau correspondant le plus à nos critères. Le problème a été séparé en deux points, le réalisme des mouvements et la prise en compte de l'autre. On peut voir sur les images d'exemples de trajectoires que nous avons obtenus des courbes plutôt réalistes, notamment grâce à la loi de Viviani, et que le CTRNN s'est montré capable de prendre en compte les déplacements de l'humain. Une version du CTRNN incorporant ces deux caractéristiques à été testée lors d'expérimentations et l'étude des résultats à révélé une nette amélioration par rapport aux algorithmes précédents.

Mais la configuration du CTRNN n'est pas encore parfaite, le grand nombre de critères de sélections entraine certain de ces critères à dominer les autres. Une solution consisterait à utiliser un apprentissage incrémental avec l'algorithme génétique. Il faudrait entraîner une population de réseaux avec seulement les critères de réalisme des mouvements, pour apprendre au réseau à bouger comme un humain. Puis prendre cette population et s'en servir de population initiale dans un nouvel algorithme génétique compor-

tant cette fois les critères de prise en compte de l'autre, pour lui apprendre à interagir. Un peu comme on apprend à dessiner avant d'apprendre à écrire.

D'autres pistes, comme étudier l'évolution de la distance des curseurs de deux humains pendant l'interaction pour reproduire cette dynamique avec le CTRNN, seront creusées d'ici la fin du stage.

Références

[dGSC10] Ben Goertzel ; Ruiting Lian ; Itamar Arel ; Hugo de Garis ; Shuo Chen. A world survey of artificial brain projects, part ii : Biologically inspired cognitive architectures. *Neurocomputing*, 74 :30–49, December 2010.

[Goe09] B. Goertzel. Opencog prime : a cognitive synergy based architecture for embodied artificial general intelligence. In *Proceedings of the ICCI-09, Hong Kong*, 2009.

[Ike09] Jean-Julien Aucouturier ; Takashi Ikegami. The illusion of agency : Two engineering approaches to compromise autonomy and reactivity in an artificial system. *Adaptative Behavior*, 17(5) :402–420, October 2009.

[JNBL03] Andrew C. Beall Jeremy N. Bailenson, Jim Blascovich and Jack M. Loomis. Interpersonal distance in immersive virtual environments. *PSPB*, 29(7) :819–833, July 2003.

[Man11] Kristen Manac'h. *Vers la notion d'agent nactif virtuel Application l'approche dynamique volutionnaire*. PhD thesis, 2011.

[MS00] Anthony Steed Mel Slater. A virtual presence counter. *Presence : Teleoperators and Virtual Environments*, 9(5) :413–434, October 2000.

[Nis07] Yasser F. O. Mohammad ; Toyoaki Nishida. Intention through interaction : Toward mutual intention in real world interactions. *New Trends in Applied Artificial Intelligence*, 4570 :115–125, 2007.

[Pen10] Ben Goertzel ; Itamar Arel ; Cassio Pennachin. Cogbot an integrative cognitive architecture aimed at emulating early childhood intelligence in a humanoid robot. 2010.

[PV91] Roland Schneider Paolo Viviani. A developmental study of the relationship between geometry and kinematics in drawing movements. *Journal of Experimental Psychology : Human Perception and Performance*, 17 :198–218, 1991.

[San07] David Vernon ; Senior Member ; IEEE ; Giorgio Metta ; Giulio Sandini. A survey of artificial cognitive systems : Implications for the autonomous development of mental capabilities in computational

agents. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 11(2) :151–180, april 2007.

[Tis09] Pierre De Loor ; Kristen Manac'h ; Jacques Tisseau. Enaction based artificial intelligence : What about human in the loop. *Minds and Machines*, 19 :319–343, 2009.

[Tis11] Pierre De Loor ; Kristen Manach ; Cyril Bossard ; Jaques Tisseau. Consideration phenomenologique et enactive de lintercomprehension humain-systeme : La piste des systemes dynamiques autonomes. 2011.

[Val12] Teddy Vallée. Application de méthodes d'analyse dans le cas d'une interaction minimaliste impliquant humains et agents. Master's thesis, ENIB, 2011-2012.