Rapport de projet de Réalité Augmentée

Robot-Sapiens

François Suro
Thibaut Castanié
Vincent BAZIA
M2 IMAGINA

18 Décembre 2015

Table des matières

1 Description du projet

Robot-Sapiens est un projet que nous avons développé dans le cadre des modules "Vision et Réalité Augmentée" ainsi que "Sociétés virtuelles".

L'objectif de ce projet est de mettre en place des robots capables d'apprendre par eux même. Dans notre cas, d'apprendre à se déplacer avec aise dans un environnement virtuel composé d'obstacles que les robots devront apprendre à éviter un maximum.

A ce projet, nous avons intégré deux outils afin d'interagir avec notre environnement virtuel.

Le Leam Motion qui, par la détection des mouvements des mains, nous permet de placer et déplacer différents types d'obstacles.

Le Track IR permet, grâce à la détection de l'orientation de la tête, de gérer la rotation de la caméra.

1.1 Le système d'apprentissage des robots

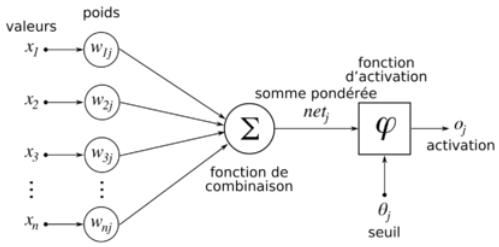
L'architecture de notre projet est divisée en deux éléments principaux : Un serveur Java et un client Unity communiquant entre eux grâce aux sockets.

Le serveur Java est le "cerveau" des robots. Son objectif est d'analyser les informations des capteurs que lui envoie le client et d'y répondre par les actions à effectuer.

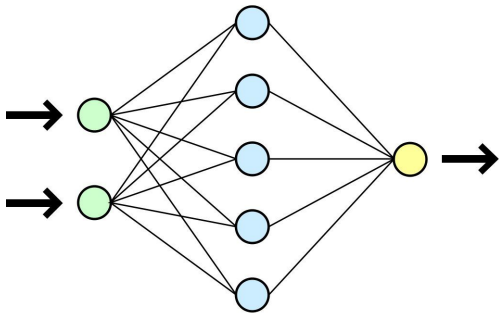
Le cerveau son apprentissage fonctionne selon deux principes complémentaires.

1.1.1 Réseau de neurones artificiel

Pour la prise de décision, nous utilisons un réseau de neurones artificiel.



Tout d'abord, un neurone artificiel est un élément qui, à de multiples entrées, attribue un résultat. Pour cela, il applique à chacune de ses entrées un poids. Il fait ensuite la somme de ses entrées pondérées puis la seuil. C'est le résultat de sortie du neurone.



Un réseau de neurones est donc une architecture en couches de neurones. Il y a trois types de couches différentes.

La couche d'entrée correspond aux valeurs entrantes des capteurs de distances du robot.

La couche de sortie est l'ensemble des résultats à cette entrée, et ses valeurs seront appliquées aux roues du robot.

Puis, une ou plusieurs couches cachées servent de transitions entre l'entrée et la sortie.

Dans un réseau de neurones, nous modifions les poids des synapses des neurones. C'est ce qui va modifier les comportements face à l'environnement. Ainsi, à force d'équilibrage de ces poids, nous obtiendrons des comportements de plus en plus adaptés.

Justement, équilibrer ces poids est le travail de l'algorithme génétique.

1.1.2 Algorithme génétique

L'algorithme génétique a pour but de générer des robots de plus en plus performants.

À l'état initial, tous les poids des synapses des réseaux de neurones sont aléatoires. Bien évidemment, les comportements de nos robots seront donc relativement mauvais. Mais tant bien que mal, certains seront plus performants que d'autres (parcours plus de distance, évite un peu mieux les murs).

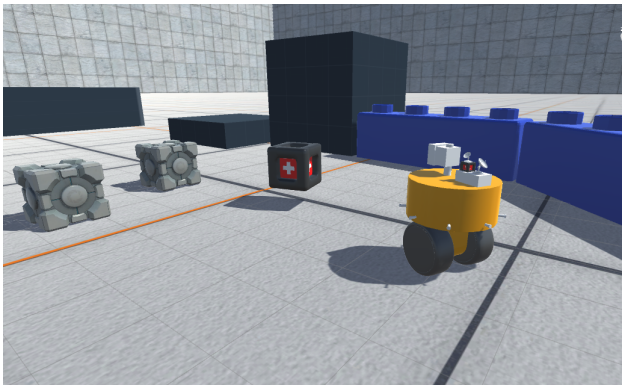
L'objectif de l'algorithme génétique est donc de classer par performance ces comportements et de générer la prochaine génération par brassage et mutation des meilleurs. Au fur et à mesure, à force de conserver les meilleurs comportements, on va obtenir des robots de plus en plus adaptés à leur environnement.

Dans notre cas, le brassage est la combinaison des poids des réseaux de neurones de deux comportements. Tandis que la mutation est une légère modification aléatoire des poids du réseau de neurones.

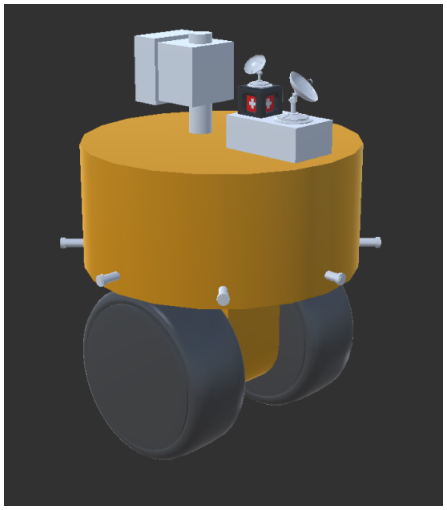
1.2 Client Unity

Communiquant avec le serveur, notre client Unity est l'environnement de travail de nos robots.

Cet environnement est composé de blocs statiques, mais aussi d'éléments dynamiques que l'on va pouvoir ajouter et déplacer tel que les cubes "LEGO", "Portal" ou encore les "packs de vie" (objectif que les robots doivent récupérer).



Nos robots sont très simples. Ils possèdent deux roues pour se déplacer, huit capteurs de distances par raytracing qui entourent le robot, ainsi qu'un capteur de distance avec les "packs de vie". Les valeurs des capteurs sont la couche d'entrée de nos réseaux de neurones, tandis que les valeurs de la couche de sortie sont affectées aux roues du robot.



1.3 Dispositifs d'interactions

1.3.1 Leap Motion

Le *Leap Motion* est un dispositif de reconnaissance de mouvement des mains, pour la réalité virtuelle. C'est un périphérique USB qui est fait pour être placé sur un bureau, face vers le haut, entre l'écran et l'utilisateur.

Il détecte une zone, à peu près hémisphérique, d'une taille d'environ un mètre, grâce à deux caméras thermiques et trois LED infrarouges. Les caméras génèrent 300 images par secondes de données qui sont analysées par le logiciel du contrôleur.

La méthode d'analyse n'a pas été divulguée par la société, mais il s'agit certainement de géométrie épipolaire utilisée pour récupérer des données de position 3D en comparant les images 2D générées par les deux caméras.

Pour intégrer ce système dans Unity, nous avons utilisé le SDK compatible avec Unity.

Ensuite, nous avons utilisé un modèle graphique de main, possédant un squelette, sur lequel est reporté le mouvement des mains de l'utilisateur. Cette technique permet de détecter facilement les collisions entre le modèle de main et les objets de la scène, en utilisant le moteur physique de Unity.

Enfin, pour détecter les gestes des mains, nous capturons les images données par le *Leap Motion*. Ensuite, on peut compter les doigts détectés (donc "ouverts"), afin de savoir si la paume de l'utilisateur est ouverte ou fermée. Il est aussi possible de détecter des gestes préprogrammés dans le SDK du contrôleur, tels que le *swipe* vers la gauche ou vers la droite, ou encore un *tap* avec l'index.

Interaction avec les blocs : `./Robot-Sapiens/.../cubeInteraction.cs`
Gestion des états : `./Robot-Sapiens/.../leapStatesManagement.cs`

1.3.2 Track IR

Le *Track IR* est un dispositif permettant de récupérer les mouvements de tête de l'utilisateur. La position et l'orientation de la tête sont calculés par une caméra placée sur le moniteur de l'utilisateur. Cette caméra suit une lumière infrarouge invisible émise par un émetteur équipé sur la tête de l'utilisateur.

Pour son implémentation, nous avons utilisé un plugin Unity développé par Tobias Boogh. Ce plugin est composé d'un dll afin de gérer le composant physique du Track IR, ainsi que d'une classe TrackIrClient pour récupérer les informations de position et de rotation du composant.

Par la suite, on détecte dans Unity si le Track IR est fonctionnel. Si c'est le cas, on substitue le contrôle souris de la caméra du FPSController par le Track Ir.

Enfin, on vérifie l'état du Track IR pour le remplacer ou non par le contrôle par souris. Mais aussi, on gère aussi les micros-coupures entre le capteur et l'émetteur pour supprimer les sursauts de caméra.

Implémentation du TrackIR : `./Robot-Sapiens/.../TrackIRManager.cs`

2 Manuel d'utilisation

2.1 Lancement

Notre projet est donc divisé en deux parties. Un serveur Java et un Client Unity.

Tout d'abord, il faut lancer le Serveur Java avec le port 28001 disponible. Pour régénérer un cerveau vierge, il faut supprimer tous les dossiers contenus dans le dossier "botBrain".

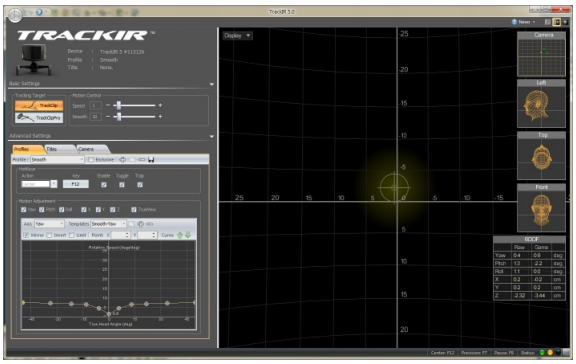
Lancer le projet Unity.

2.2 Utilisation du Track IR

Le Track IR est un outil qui permet de traquer la position et l'orientation de la tête de l'utilisateur.

Dans un premier temps, il faut placer le capteur sur le moniteur orienté face à l'utilisateur.

Ensuite, il faut équiper l'émetteur sur la tête de l'utilisateur par le biais d'un casque audio, d'une casquette...



Une fois bien positionner face à l'écran, il faut calibrer l'état initial du positionnement de la tête de l'utilisateur par le biaisi de l'interface de gestion du Track IR.

Dans notre projet, seule l'orientation du visage est prise en compte et non sa position. Orienter son visage dans une direction fait tourner la caméra dans ce sens.

2.3 Leap Motion

Le Leap Motion est un outil qui permet de détecter le mouvement des mains ainsi que la disposition des doigts.

Nous avons défini différentes actions possibles :

- Le poing fermé permet d'ouvrir, ou de fermer en validant, le menu
- Un mouvement de la main vers la gauche ou la droite permet de naviguer dans le menu des objets à déposer
- Le *tap* avec l'index met un fantôme de l'objet dans la main virtuelle, et permet son déplacement, sans soucis de collision. Un nouveau *tap* avec l'index instantie l'objet à la position choisie précédemment
- La paume ouverte, permet d'attraper des objets en entrant en collision avec eux. Une fois l'objet en main, il devient un fantôme :
  - Un *tap* pour le "relâcher" à la position du fantôme
  - Fermer le poing repose l'objet à sa place initiale (annulation)
- L'index pointé permet de supprimer des objets avec lequel il entre en collision

3 Caractéristiques du projet

Nous sommes fiers d'avoir réussi à développer une architecture permettant l'apprentissage de robots virtuels. L'implémentation d'une telle structure était ambitieuse, mais nous sommes parvenus à avoir un programme fini, qui fonctionne très bien.

Nous avons réussi à utiliser deux dispositifs d'intégrations différents, tout en conservant une bonne ergonomie pour l'utilisateur. Le *trackIR* permet de contrôler la caméra et la direction sans utiliser les mains, le *Leap Motion* permet d'agir sur l'environnement en utilisant une seule main. Enfin, la dernière main disponible est utilisée pour se déplacer dans la scène, au clavier. L'adaptation à une telle interface se fait instantanément et l'utilisateur n'est pas dérouté.

4 Difficultés rencontrées

Nous avions dans l'idée d'utiliser l'Oculus Rift à la place du Track IR.

Seulement, on a eu des problèmes dans l'implémentation de l'Oculus Rift DK1 présent au département informatique. Il ne fonctionnait pas correctement sur nos ordinateurs et il était impossible d'utiliser le SDK en raison d'incompatibilité dans les versions.

Nous ne pouvions pas non plus utiliser d'alternatives à base de smartphone, tel que la Google Cardboard puisque notre projet est une architecture complexe composé d'un serveur Java et d'une client Unity.

C'est pour cette raison que nous nous sommes intéressés au Track IR, même si la sensation n'est pas autant immersive et impressionnante. Elle fonctionne avec notre projet et permet de limiter un maximum l'utilisation de périphériques classiques, tels le clavier et la souris.

5 Références

- Réseau Neuronal simple, codé par **Matthew Robbins**, en C++
  `https://github.com/matthewrdev/Neural-Network`
- Gestion du Leap Motion sur Unity
  `https://developer.leapmotion.com`
- Plugin Unity Track IR, codé par **Tobias Boogh**
  `https://github.com/byBrick/Unity-TrackIR-Plugin`

Annexes

- Dépot GitHub du Projet : `https://github.com/Ooya/Robot-Sapiens`
- Le site du Leap Motion : `https://www.leapmotion.com/`
- Le TrackIR, par NaturalPoint : `https://www.naturalpoint.com/trackir/`