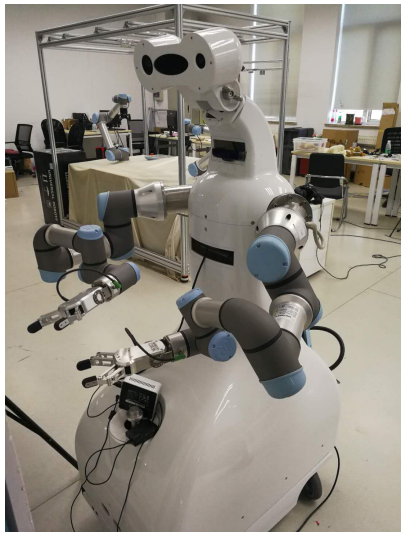
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

YOU Yang

MÉMOIRE DE STAGE DE SPÉCIALITÉ

Stage de recherche robotique



Stage effectué du 28/05/2018 au 16/09/2018

CASIA

Maître de stage : **Dr. XU Bo**

Remerciements

Tout à bord, je voudrai remercier à Dr. XU Bo pour me donne une opportunité de stage comme un chercheur dans la domaine robotique. Il est très patience et il m'a guidé beaucoup pendant mon stage en CASIA.

Résumé

Durant le stage à CASIA, mon travail principal est rechercher un robot avec deux bras pris un objet avec deux bras en collaborant. Il contient trois parties de travail, la première partie du stage est conception mécanique pour le robot, il faut créer les modèles d'après le robot réel afin de pouvoir simuler avec l'algorithme dans l'environnement physique virtuelle. La deuxième partie de stage est étudier le « Robot Learning », il contient un grand nombre des équations mathématique, la méthode plus importante dans cette partie est « Deep Reinforcement Learning ». La troisième partie aussi le plus important et difficile mission est utiliser « Deep Reinforcement Learning », améliorer et créer les nouvelles méthodes des algorithmes pour mettre le robot avec deux bras pris un objet avec deux bras en collaborant.

1.Introduction d’entreprise

1.1 Introduction de l'Institut d'automatique de l'Académie chinoise de sciences (CASIA)

L’Académie chinoise de science: L’Académie chinoise de science est l’académie nationale pour les sciences naturelles de la Chine． Elle dépend du Conseil des affaires de l’Etat et son siège à Beijing. L’Institut d’automatique de l’Académie chinoise de sciences (CASIA) est un institut spécialisé à les domaines automation robotique et Intelligence artificiel.

1.2 Relation avec la France

Le laboratoire franco-chinois de recherche en informatique, automatique et mathématiques appliquées (LIAMA) est un laboratoire de recherche sino-français créé en 1997 par l'Académie chinoise des sciences et l'Institut national de recherche en informatique et en automatique. Le LIAMA est hébergé par le CASIA, l'Institut d'automatique de l'Académie chinoise de sciences, à Pékin, près du technopôle Zhongguancun. Les projets sont réalisés en coopération entre l'Europe et la Chine et peuvent être hébergés par toute institution partenaire.

Le LIAMA a été désigné par le ministère chinois de la Recherche comme « centre national pour la recherche internationale » en juin 2008 afin de renforcer les coopérations potentielles en informatique.

Le LIAMA est géré par un consortium de partenaires qui a été fondé en octobre 2008.

Le LIAMA est composé d'une équipe de 120 chercheurs permanents, associés et d'étudiants chinois, français mais aussi d'autres pays européens.

1.3 Laboratoire complexé et science intelligent

Mon stage est effectué dans Laboratoire complexé et science intelligent, C’est un laboratoire qui recherche l’utilisation de Intelligence Artificiel dans le robot pour résoudre les problèmes complexés.

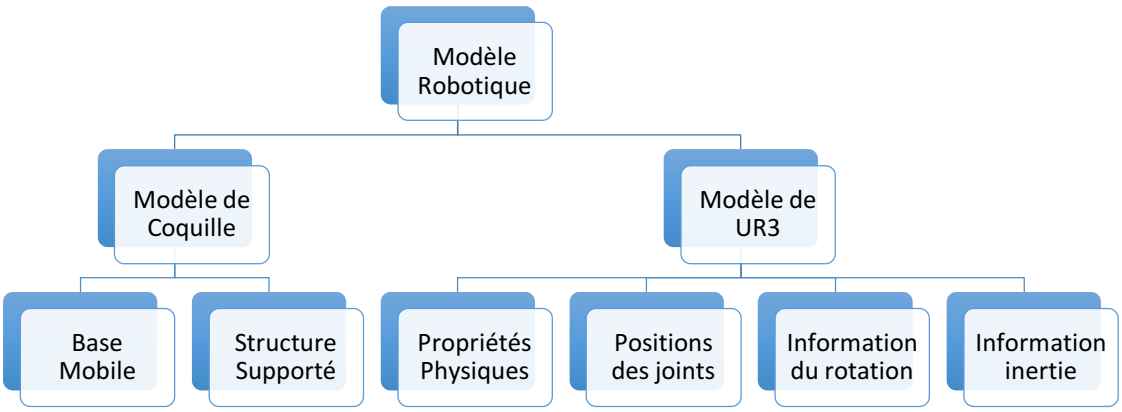
2.Partie 1 : Modèle et Environnement

2.1 Modélisation des pièces robotiques

La modélisation est fondamentale dans la recherche, Il faut crée tous les modèles des équipements nous avons utilisée, nous avons utilisé les équipements comme ci-dessous :

- Universal Robot UR3 *2
- Coquille
- Kinect V2
- Radar Laser

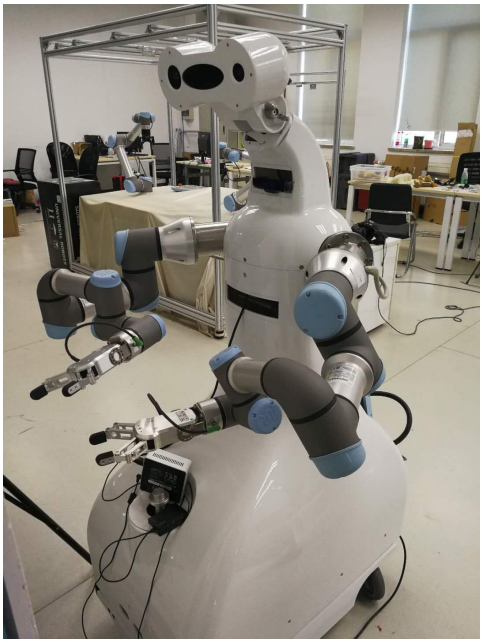
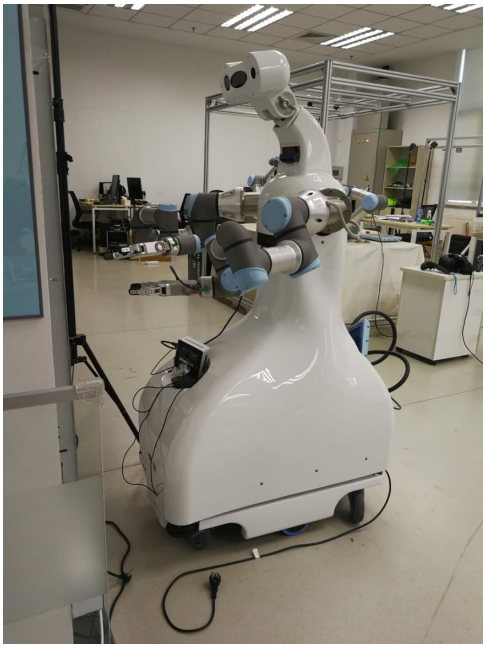
Le processus de modélisation est présenté comme ci-dessous :



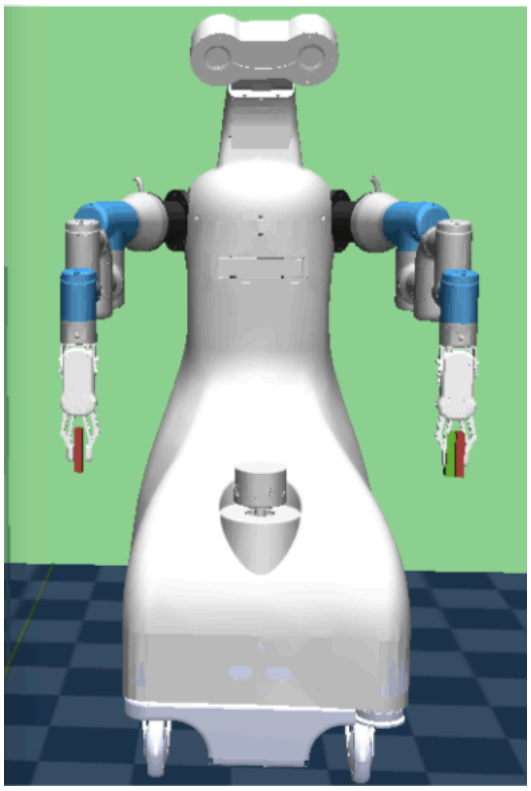
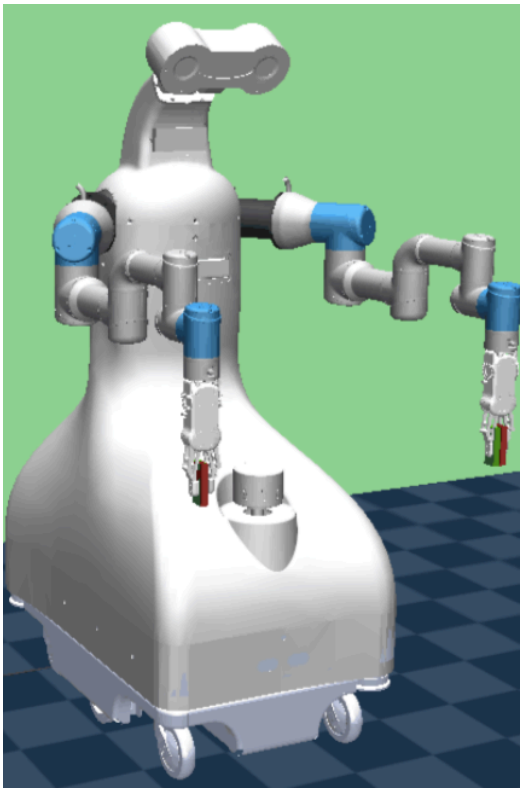
2.2 Assemblage des pièces robotiques

Après la création des pièces, ensuite nous besoin assembler les pièces ensemble d'après notre robot.

Notre robot après constitution est placé comme ci-dessous :

Notre modèle après conception mécanique et assemblage est donné comme ci-dessous :

Après l'assemblage, il faut transférer les ficher(.STL) dans la environnement physique, mais il n'est pas encore suffisent pour lancer la simulation jusqu'à maintenant, il

manque la définition de propriété physique pour chaque partie du robot comme la masse, inertiel, position, axis et dégrée de liberté etc.

2.3 Définition des propriétés physiques dans la ficher XML

Dans l'environnement physique virtuelle (Mujoco), nous besoin définir les propriétés des objets dans un certain ficher(XML), c'est un type de ficher qui introduire les fonctionnements des chaque partie du modèle, il est très similaire à .HTML qui est le ficher du front-end d'application Web.
Un exemple de la structure de .XML est présenté comme ci-dessous :

```
<mujoco>
    <default class="main">
        <geom rgba="1 0 0 1"/>
        <default class="sub">
            <geom rgba="0 1 0 1"/>
        </default>
    </default>

    <worldbody>
        <geom type="box"/>
        <body childclass="sub">
            <geom type="ellipsoid"/>
            <geom type="sphere" rgba="0 0 1 0"/>
            <geom type="cylinder" class="main"/>
        </geom>
    </worldbody>
</mujoco>
```

Dans cet exemple, d'abord il y a deux class par défaut, « main » et « sub ». Ensuite, dans la partie « worldbody », il utilise un grand type « box » qui inclue trois géométries dedans.

Après la définition de structure, il faut ajouter les propriétés physiques pour chaque géométrie, Ici c'est un exemple contient la position relative, la masse, l'inertie et information de rotation.

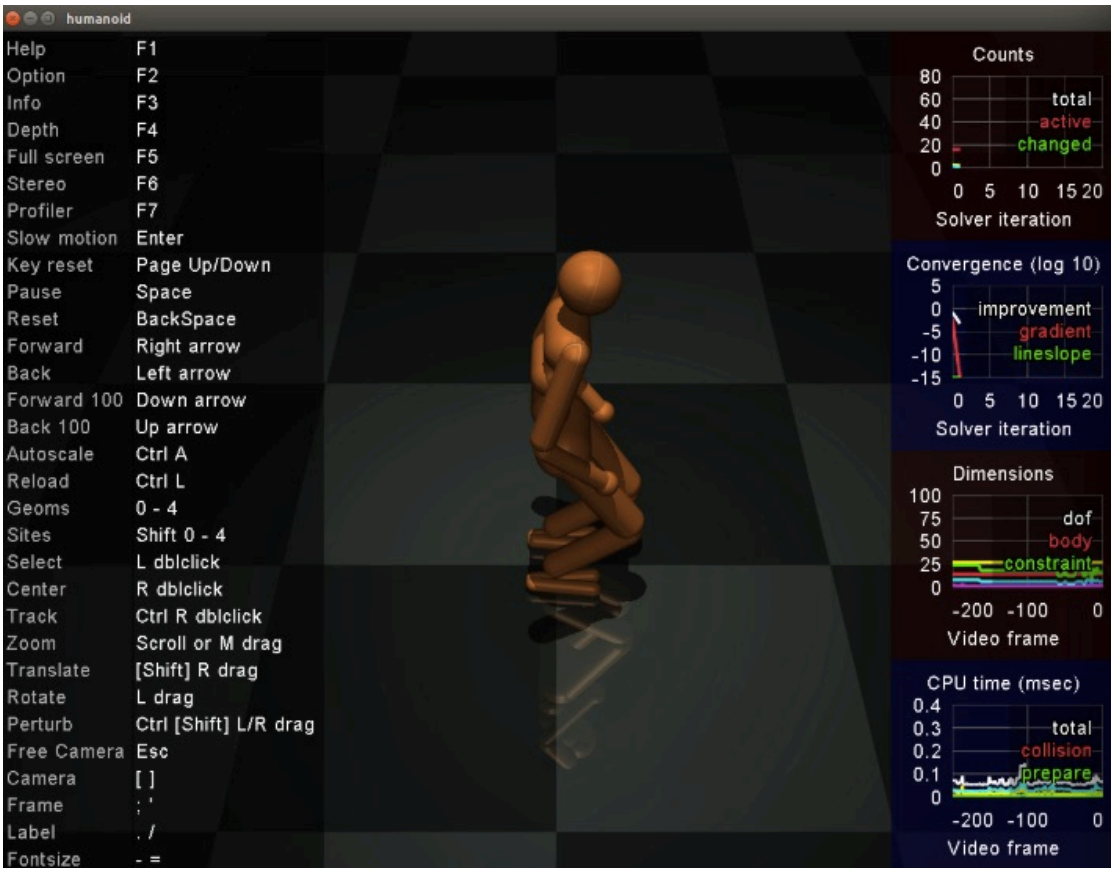
```
<body>
   <geom type="box" pos="1 0 0" size="0.5 0.5 0.5"/>
</body>
```

```
<body pos="1 0 0">
   <inertial pos="0 0 0" mass="1000" diaginertia="166.667 166.667 166.667"/>
   <geom type="box" pos="0 0 0" size="0.5 0.5 0.5"/>
</body>
```

2.4 Environnement virtuel physique (Mujoco)

Maintenant il y a deux environnement virtuel physique populaire : Gazebo et Mujoco, normalement il utilise Gazebo dans ROS (Robot Operating system). Durant mon stage, nous avons utilisé Mujoco.

Introduction du Mujoco:



un image de Mujoco.

Le nom total de Mujoco est « Multi-Joint dynamics with Contact ». C'est un simulateur physique pour le robot, biomécanique, graphique et animation, apprentissage automatique etc. Il est rapide et précis dans la simulation, il est supporté à calcul parallèle, qui est très important quand il y a des simulations complexées.

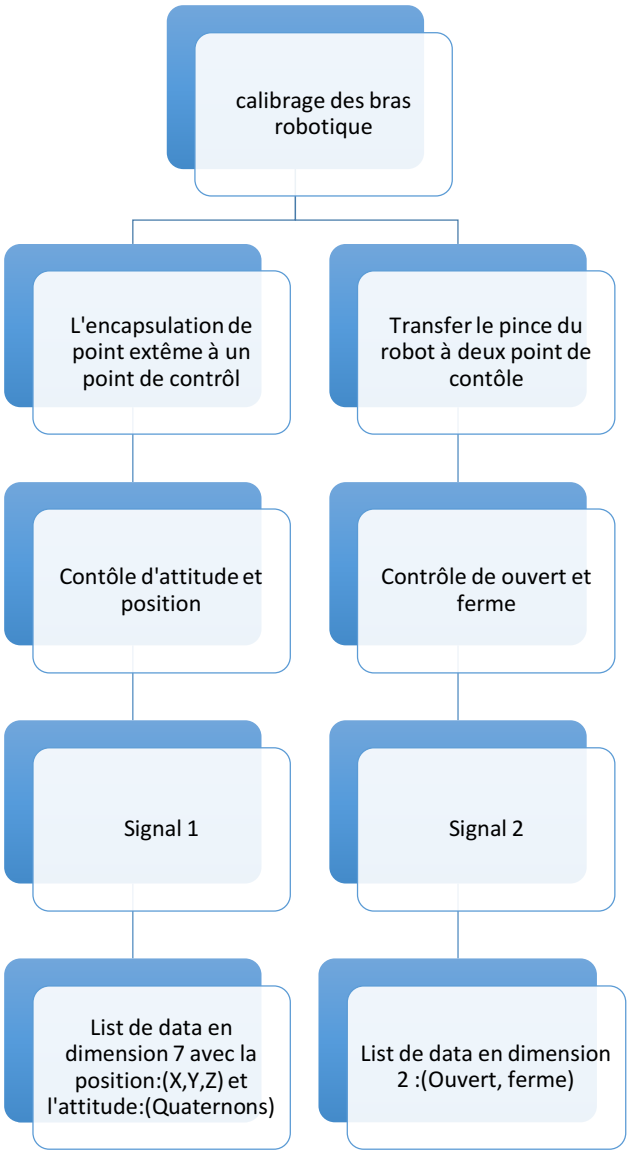
Les techniques utilisées dans Mujoco sont données comme ci-dessous :

	High level	Low level
File	MJCF/URDF (XML)	MJB (binary)
Memory	mjCModel (C++ class)	mjModel (C struct)

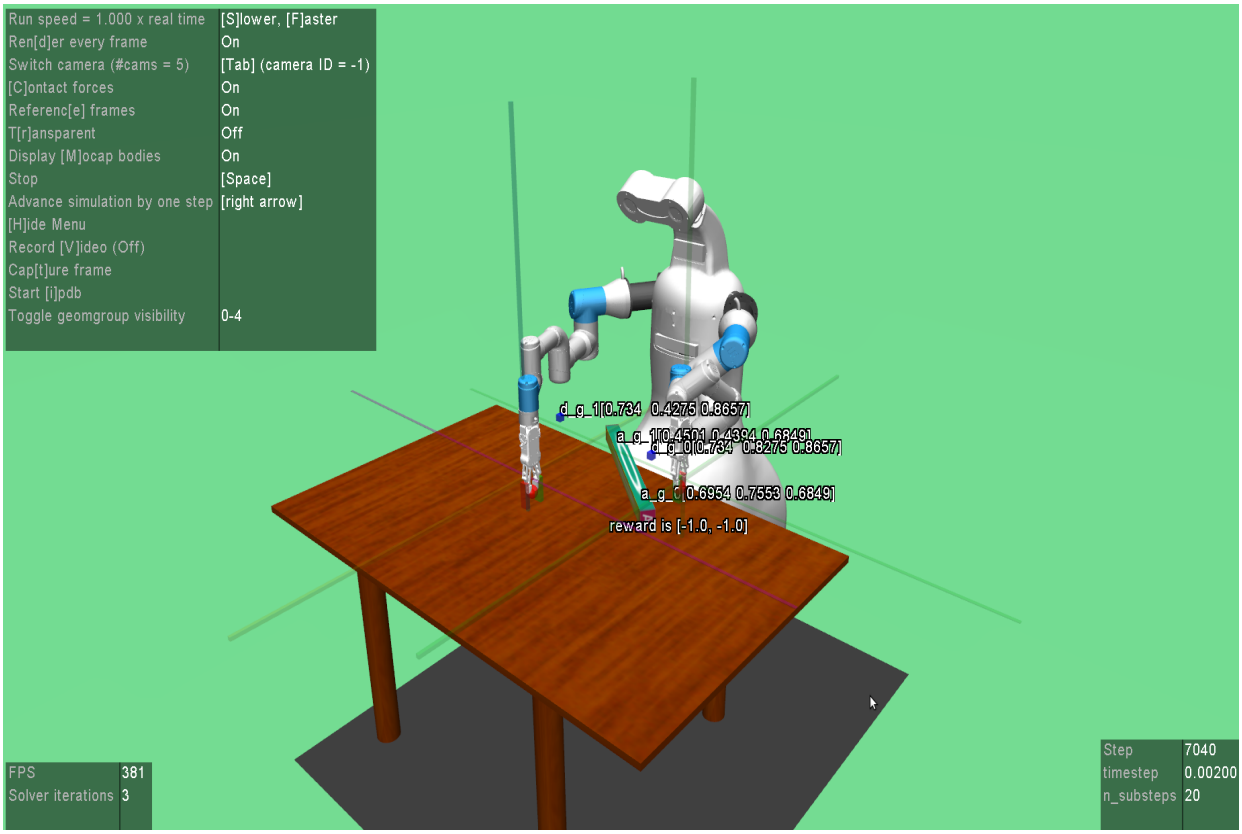
Mujoco est écrit par langage C++, mais il y a une interface de python (mujoco-py) qui permet utiliser les codes python pour tester les algorithmes dans le simulateur physique.

2.5 Intégration et simulation sur Mujoco

Après l'installation de Mujoco et introduire les modèles du robot, les fiches .XML, nous pouvons intégrer et simuler notre model dans Mujoco, d'abord, on besoin crée un programme qui généré une destination stochastique, cette position va destiner la coordonné du point extrême du bras robotique. Une structure de contrôle est présentée comme ci-dessous :



Un exemple est placé comme ci-dessous :



Dans cet exemple, nous avons créé un environnement qui contient une table avec un Object rectangle en dessus, nous avons programmé pour l'affichage les positions du gauche et droite des objet, les coordonnées des deux points extrêmes du bras robotique. Nous avons fixé deux axis aux les deux points extrêmes du bras robotique, dans la recherche future, nous avons étudié l'utilisation des algorithmes pour les deux axis peut arriver aux deux points du destination dessiné.

Puis, dans ce test, nous avons vérifié est ce que notre modèle du robot est bien crée et assemblé. S'il y a deux jeux entre les pièces, nous pouvons le visualiser dans ce test.

Il faut lancer le test à chaque fois s'il y a un changement du fichier .XML, parce que quelque fois il y a des problèmes à cause de la collision, qui est un résultat de mal définition du range de joint ou la position relative des pièces dans la ficher .XML du modèle.

3.Partie 2 : Robot Learning

3.1 Introduction du « Robot Learning » et « Deep Reinforcement learning »

Le « Robot Learning » est un domaine de recherche à l'intersection de l'apprentissage automatique et de la robotique. Il étudie des techniques permettant à un robot

d'acquérir de nouvelles compétences ou de s'adapter à son environnement grâce à des algorithmes d'apprentissage. Le mode de réalisation du robot, situé dans une incorporation physique, fournit à la fois des difficultés spécifiques (par exemple, haute dimension, contraintes en temps réel pour la collecte de données et d'apprentissage) et des possibilités de guidage du processus d'apprentissage (synergies sensori-motrices, primitives motrices).

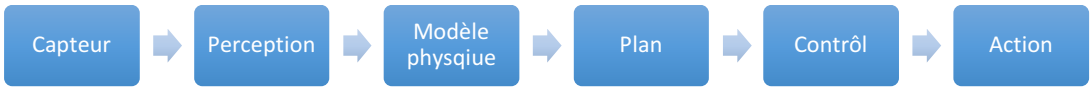
Le « Robot Learning » peut être étroitement lié au contrôle adaptatif, à le « Deep Reinforcement Learning » ainsi qu'à la robotique développementale, qui prend en compte le problème de l'acquisition autonome à vie des répertoires de compétences. Bien que l'apprentissage par machine soit fréquemment utilisé par les algorithmes de vision par ordinateur utilisés dans le contexte de la robotique, ces applications ne sont généralement pas appelées le « Robot Learning ».

En intelligence artificielle, plus précisément en apprentissage automatique, l'apprentissage par renforcement consiste(Reinforcement Learning), pour un agent autonome (robot, etc.), à apprendre les actions à prendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps. L'agent est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, un comportement décisionnel (appelé stratégie ou politique, et qui est une fonction associant à l'état courant l'action à exécuter) optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.

Le « Deep Reinforcement Learning » = « Deep Learning » + « Reinforcement Learning », le « Deep Learning » est une méthode classique pour les problèmes avec les data en haute dimension dans l’apprentissage automatique.

3.2 Deep Reinforcement Learning dans la domaine robotique

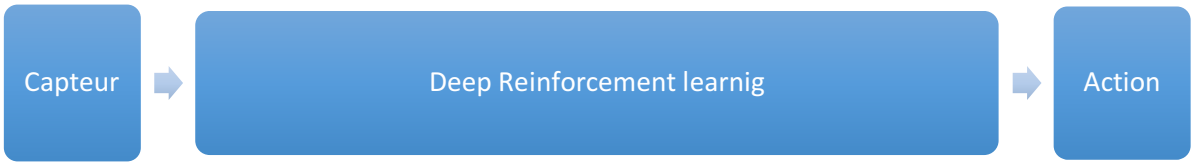
C'est une processus classique robotique.



D'abord le robot obtient les informations par les capteurs comme les images, le distance etc. Ensuite, le robot va procéder les informations et transférer à la model physique et obtenir un résultat prévu. D'après le résultat prévu, un plan du mouvement va générer. Dans la suite, les moteurs vont être contrôler afin de suivre le plan. A la fin, le mouvement ou la position finale est l'action du robot.

Dans le processus classique, il faut définir plein de règle pour les parties Perception et Plan.

Comme dans l'introduction du Deep reinforcement learning, nous pouvons remplacer tous les étapes du milieu par Deep reinforcement learning.



3.3 Alogrithmes DQN et DDPG

DQN :

DQN (Deep Q-Learning Network) est le travail de fondation dans le « Deep Reinforcement Learning », il est réussi de combiner « Deep Learning » et « Reinforcement Learning » afin de réaliser un algorithme de perception jusqu'à action.

Dans la méthode « Q-Learning », les espaces des états et actions sont discrète et basse dimensionné, il utilise un Q-Table pour stocker le Q-Value de chaque action dans son état. Le Q-Value est une valeur qui détermine la qualité de cet décision d'action.

Mais pour les problèmes haut dimensionné, c'est impossible d'utiliser Q-table dans ce cas. Pour résoudre ce problème, la méthode plus pratique est transformer le problème de mise à jour de Q-table en un problème d'ajustement de fonction qui permet il obtient les mêmes actions quand il y a des états similaires. La fonction Q est approximée à la valeur Q optimale en mettant à jour le paramètre θ.

$$Q(s,a;\theta) \approx Q'(s,a)$$

Le « Deep Neural Network » peut extraire des fonctionnalités complexes automatiquement. Il est donc préférable de l'utiliser pour les problèmes avec grande dimension.

Le DQN utilise « CNN » et « Q-Learning » ensemble, l'entrée de data(image) va aller au « CNN » et la sortie est le Q-Value de chaque action. Un perso-code de DQN est donné ci-dessous :

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t),a;\theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t,a_t,x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t,a_t,r_t,\phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j,a_j,r_j,\phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1},a';\theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j,a_j;\theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

DDPG:

Malgré la méthode DQN est préférable pour le problème en grande dimension, il n'est pas encore suffisant pour résoudre le problème avec les actions continus. Donc pour

résoudre le problème avec les actions continus et haute dimension, il faut utiliser une méthode DDPG.

Le nom total de DDPG est « Deep Deterministic Policy Gradient », il vient d'algorithme DPG (Deterministic Policy Gradient) et PG (Policy Gradient).
Pour expliquer les algorithmes, il faut d'abord voir les différentes méthodes dans « Reinforcement Learning ». En bref, le « Policy Gradient » est une méthode qui est dépend le Policy, donc il n'est dépendu pas à le modèle. Les algorithmes qui sont dépendu a les modèles sont comme Q-learning et DQN, tous les deux sont Value-Based. Donc l'entrée du Policy Network est State, et sa sortie est action, cependant dans le Q-learning, sa sortie est le Q-value.
Le perso-code du DDPG est présenté comme ci-dessous :

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

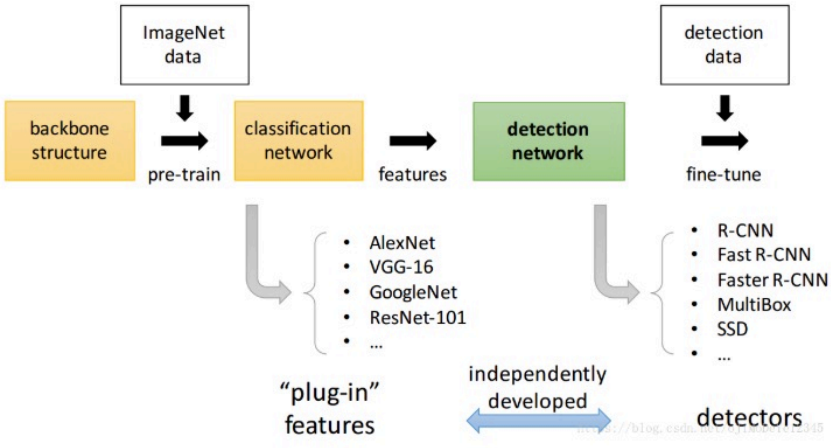
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

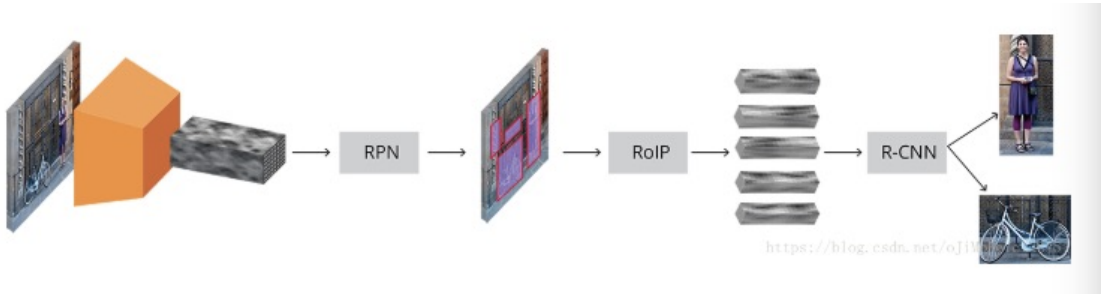
3.4 Vision Robotique

Cette partie est pour le vérification dans le robot réel dans la futur, parce que dans l'environnement virtuelle, il peut avoir la coordonnée de l'objet directement. Mais en réalité, il besoin la vision robotique pour déterminer la position de l'objet à chaque moment.

Nous avons utilisé un capteur Kinect V2 pour la détection des objets, donc il est nécessaire d'utiliser une méthode pour le réalisation la vision robotique et la détection. Voici une structure classique pour la détection des objets.

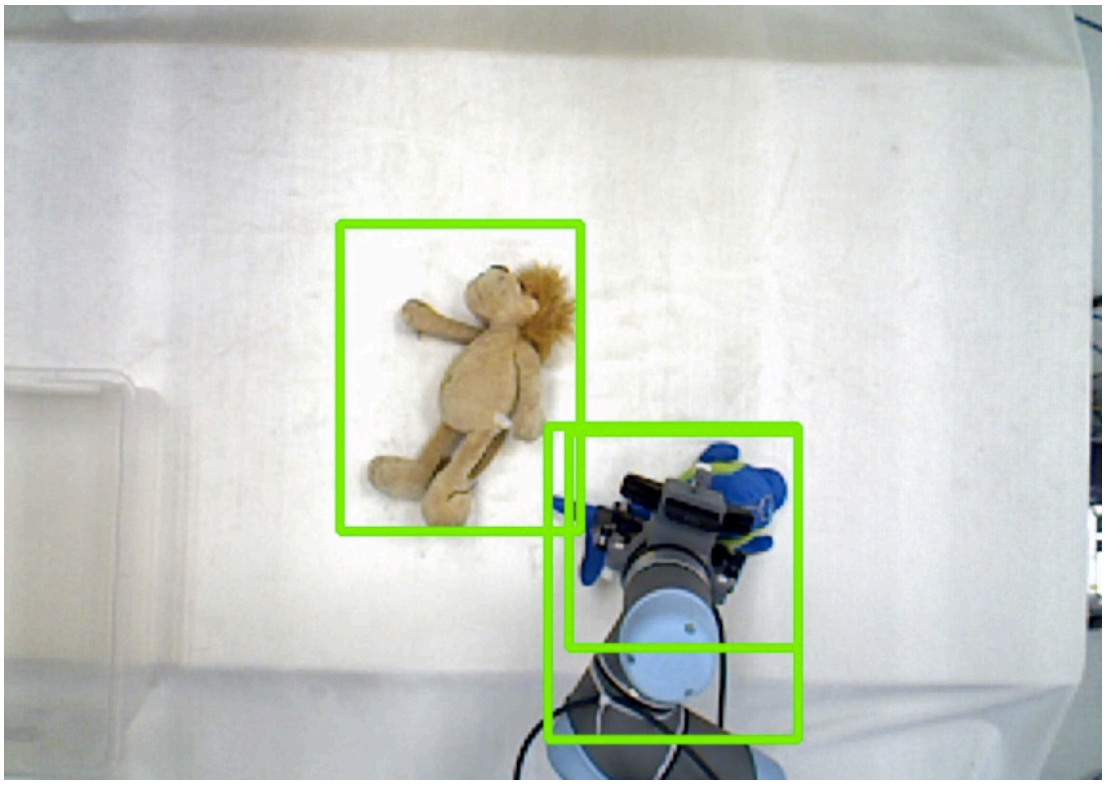


Nous avons utilisé une méthode appelé « Faster R-CNN », c'est une méthode qui vient de l'amélioration de la combinaison des RNN et CNN. La structure de « Faster R-CNN » est présenté comme ci-dessous :



D'après le capteur dans la tête du robot, nous avons pris beaucoup des photos des objets. Ensuite, nous avons classé les images par ses propriétés afin de l'apprentissage automatique dans l'étape suivant. Nous avons le divisé à deux parties, la partie d'entraînement et la partie vérifiée.

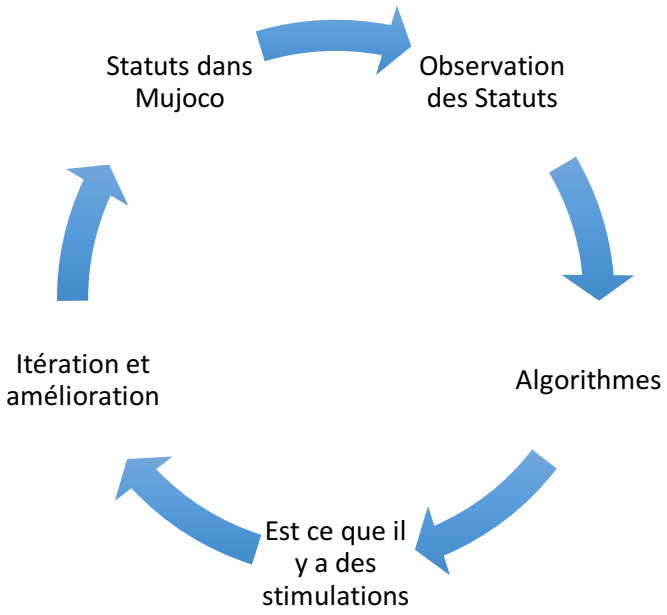
Après l'entraînement, nous avons obtient les résultats corrects qui peut classer les images et détecter les objets automatiquement. Voici un résultat nous avons eu.



4. Partie 3 – Exploration de deux bras robotique tirer un objet avec collaboration

4.1 Modification des modèles dans Mujoco

Nous avons déjà les modèles robotique, mais il n'est pas encore suffi pour lancer le test. Il faut créer les objets afin de réaliser trois missions nécessaires :
« Reach », « Push » et « Pick And Place ».
Les trois missions sont assez suffi de présenter la capabilité du robot.
Donc nous avons modifié les trois ficher (Push.xml, Reach.xml, Pick_And_Place.xml).
Nous avons aussi défini les règles des trois missions. Pour le « Reach », nous avons défini dans la programme avec les deux doigts doit bouger à deux points destinataire, les deux points sont randomisés à chaque fois ; Pour le « Push », nous avons défini le robot doit pousser l'objet à le point destinataire ;Le « Pick And Place » est le plus difficile et complexé, le robot doit prendre et maintenir l'objet à les points destinataire.
La structure d'apprentissage de mission est :

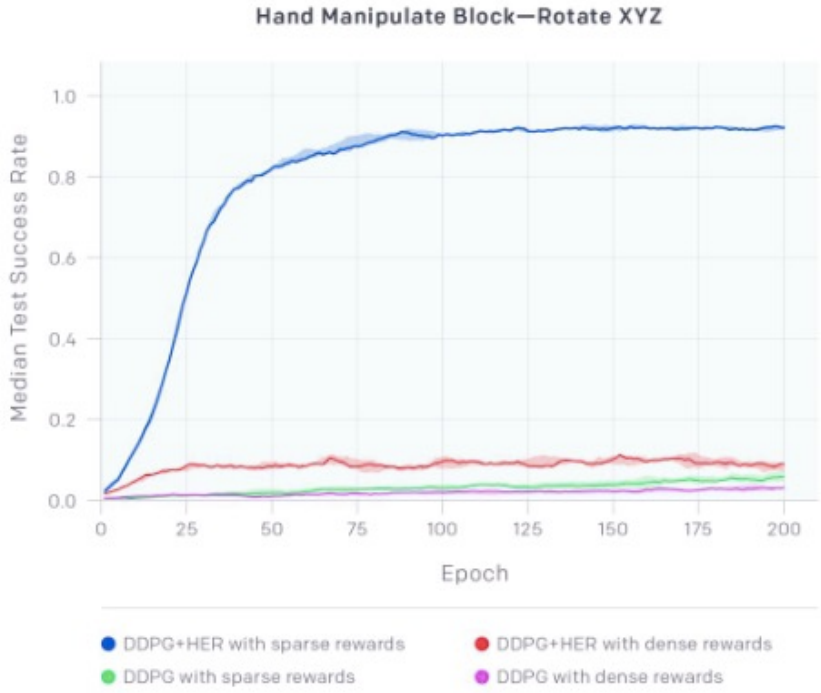


4.2 Changement des algorithmes

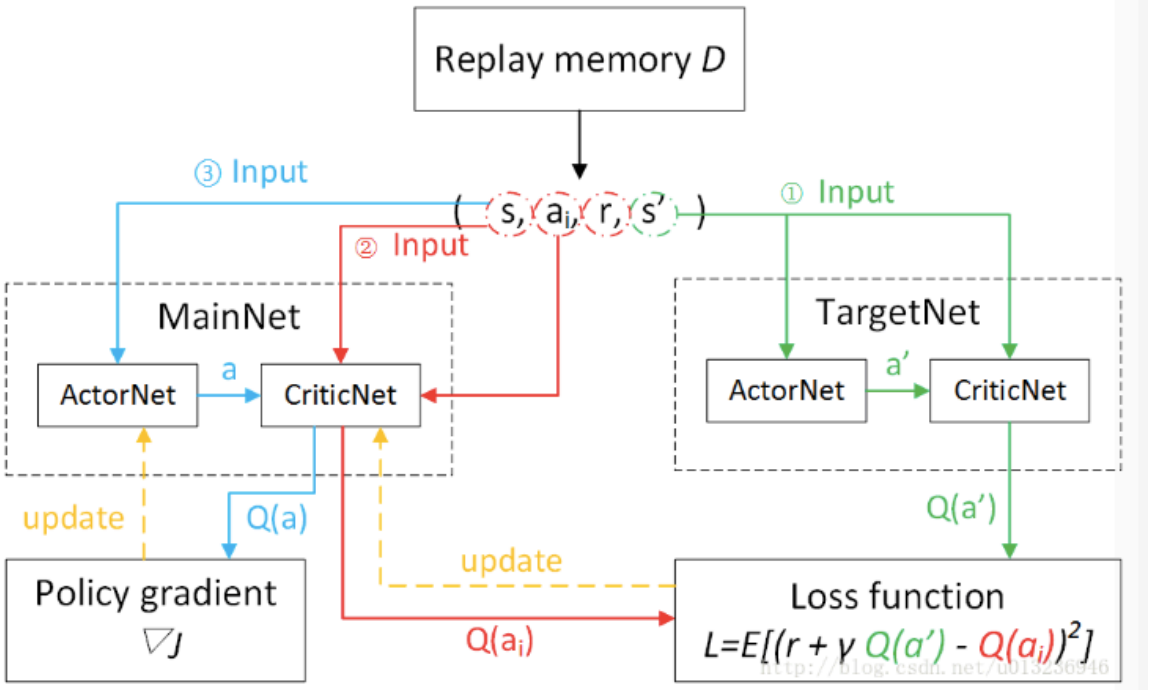
Tout à bord, nous avons utilisé DDPG dans les cas, dans les cas « Push » et « Reach », il utilise beaucoup de temps pour l'apprentissage, mais il n'a pas eu un bon résultat à la fin. Nous avons analysé la cause pour ce résultat, comme dans l'introduction de « Reinforcement Learning », l'agent peut être stimulé s'il trouve un State correct avec son Action, donc il a eu plus de possibilité d'appliqué cette méthode profondément. Mais dans notre cas, la possibilité d'avoir un résultat correct est très rare, donc les très peu stimulation sont distribué dans les espaces très haut-dimension.

En 2018, Open AI(Entreprise American) a proposé un nouveau algorithme pour résoudre le problème avec les stimulations rare, le méthode s'appelle : « Hindsight Experience Replay ». L'essentiel de « HER » est-il change de façon de stimulation, avant si l'agent n'a pas trouvé un State correct, il va perdu tous la stimulation dans cet épisode, mais dans « HER », il considère cette Action peut avoir un résultat correct si on change le destination (ou le but), donc l'agent peut avoir encore des stimulations même s'il n'a pas trouvé un bon résultat d'après la destination au début. En bref, C'est une façon qui peut garantir l'exploitation du l'environnement, et il peut apprendre l'expériences d'après les fautes.

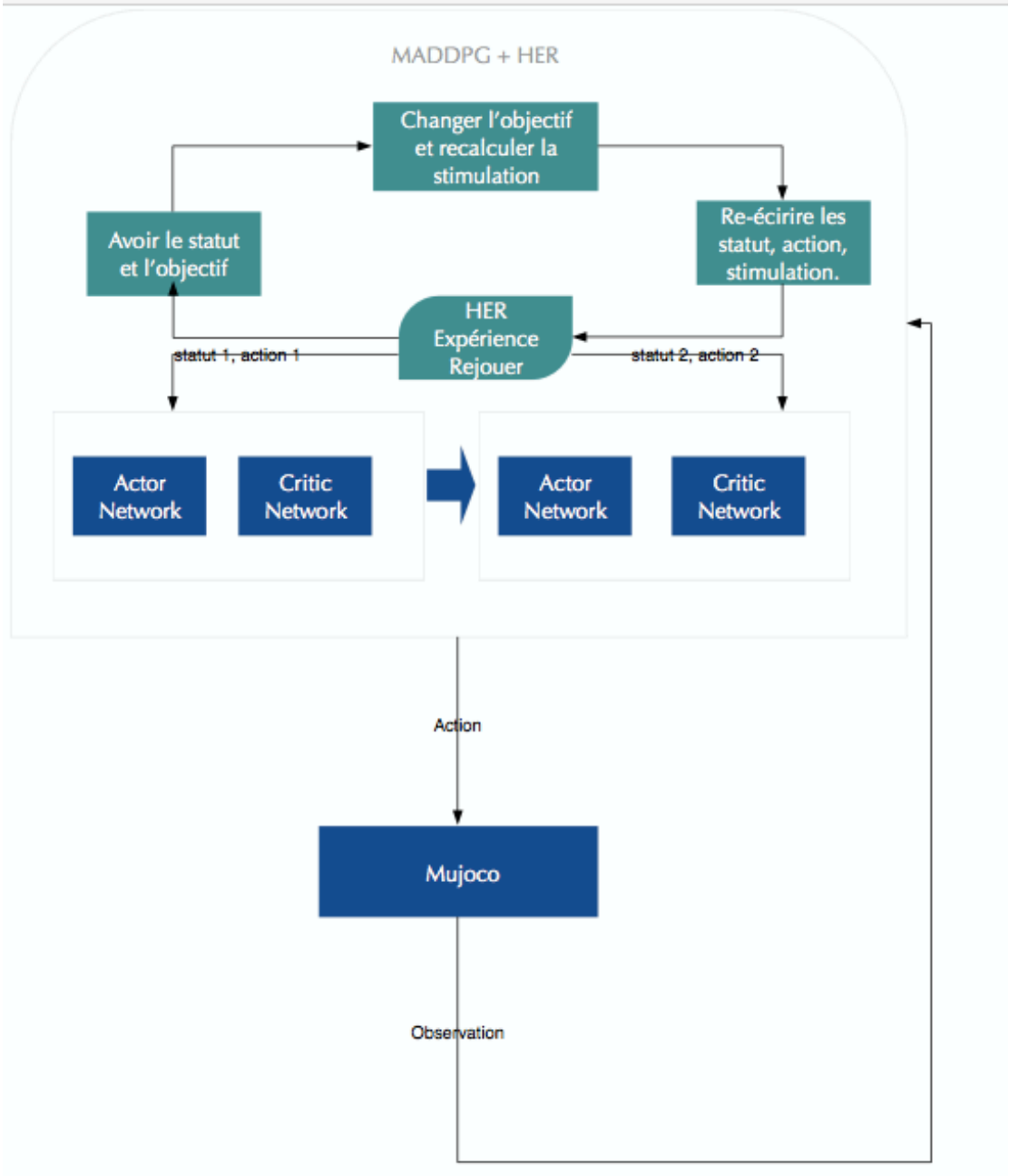
Voici un exemple classique qui présente les possibilités de la réussite.



Donc nous avons intégré DDPG et HER dans notre code, après nous avons eu les résultat beaucoup mieux dans les cas « Reach » et « Push ».
Malgré « DDPG+HER » est assez forte dans les cas précédents, mais dans le cas « Pick And Place », un problème très particulier est apparaît, les deux bras se battre pour prendre l'objet.
Pour analyser ce problème, il faut d'abord voir la structure de la méthode DDPG :

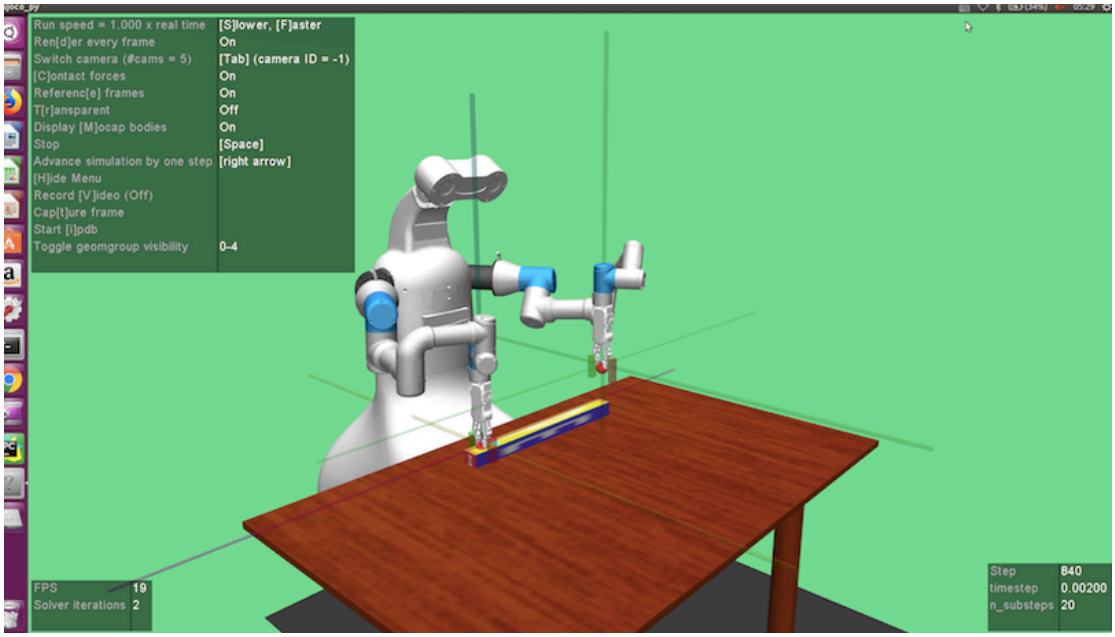


Nous pouvons voir que les informations de l'environnent (S et S') sont les entrées des Actor-Critic Network, qui guide la décision de l'agent. Mais dans notre cas, nous avons deux bras, donc ils sont deux agents séparément, les deux agents ne savent pas l'information de l'autre, c'est-à-dire, ils n'ont pas eu l'information pour coopérer. Donc les bras sont se battre sans savoir la coopération pour finir la tâche.
Ainsi la résolution de ce problème est créée une façon pour les deux bras savoir les informations de l'autre bras comme sa position et sa point destinataire.
Nous avons utilisé une méthode appelle « MADDPG », le « MA » est « Multi-Agent ».
Notre structure des algorithmes après modification est présenté ici :

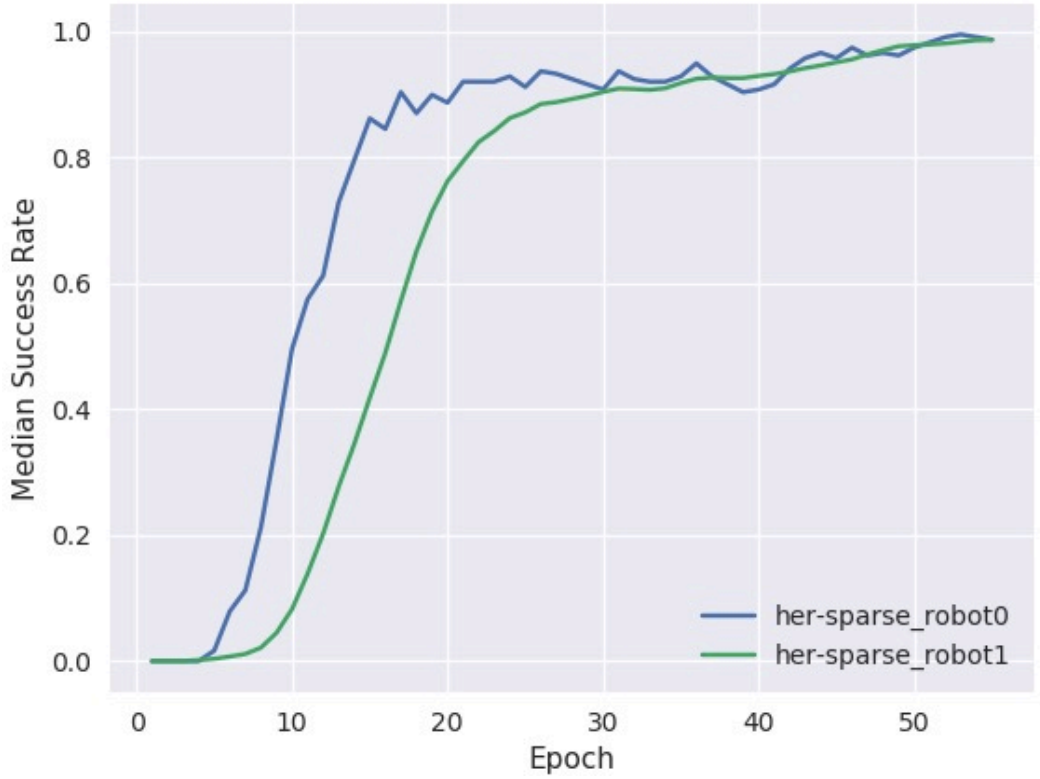


4.3 Les Résultats

Après la modification, nous avons eu les résultats beaucoup mieux, voici un exemple dans le mission « Pick And Place ».



Les possibilités de réussite d'apprentissage est donné comme ci-dessous :



Maintenant il reste un problème avec le bras 0 est beaucoup plus instable par rapport à bras 1. Notre équipe est en train de résoudre ce problème.

5.Conclusion

Durant le stage à CASIA, j'ai appris la coopération avec les autres dans la domaine robotique, notre recherche est assez avancée et très intéressant, notre équipe est en train d'améliorer les résultats et écrire un article sur ce projet, je vais être le troisième auteur avec mon travail dedans. Puis, j'ai consisté mon intérêt d'appliquer pour un PhD dans la domaine robotique, c'est le domaine j'aime vraiment beaucoup avec une grande passion.