Projet Checkers Cobot
Environnement de travail

Version n°1 - MAJ 02/02/2017

Auteurs : Skiada Alexandre, Meszaros Gordon

Résumé : On retrouvera dans ce document un ensemble de consigne permettant de faciliter la mise en place d'un environnement de travail.

1 Ressources

Voici une liste des principaux éléments de l'environnement de travail :
- une machine linux Ubuntu 14.04 LTS (ROS VM- `http://nootrix.com/downloads/#RosVM`)
- ROS Indigo + catkin
  - le Wiki ROS Universal Robot(`http://wiki.ros.org/universal_robot`)
  - le tutoriel ROS Universal Robot (`http://wiki.ros.org/universal_robot/Tutorials/GettingStartedwithaUniversalRobotandROS-Industrial`)
- Package universal_robot et son extension ur_modern_driver pour le bras
- Package wsg50-ros-pkg pour la pince

2 Installation ROS

Il a été décidé d'utiliser une machine virtuelle pour utiliser l'ensemble. En effet, l'élément principal du projet est le bras UR10 et son package associé. Il existe un tutoriel sur le wiki qui indique avoir été réalisé sur une machine virtuelle. De même, la volonté de proposer un travail réutilisable, rend la machine virtuelle adaptée.

2.1 Version Machine Virtuelle

La première étape consiste a suivre ce tutoriel (le tutoriel ROS Universal Robot) dans sa presque totalité. Dès le début, il est indiqué un lien vers la l'image .ova utilisée. Malheureusement, après de nombreux essais, il n'a pas été possible de faire fonctionner cette machine virtuelle. Il existe tout de même une machine virtuelle plus récente proposée par une tiers personne. Le principal avantage de cette dernière repose sur la version plus récente de Ubunutu (12 vs 14) ainsi que celle de ROS (Groovy vs Indigo). Plus tard, il sera nécessaire d'utiliser ur_modern_driver qu'il n'a pas été possible d'installer dans une version antérieure à Indigo.

Il vaut mieux privilégier la version 64bits qui s'est montrée beaucoup plus réactive à l'utilisation.

Une fois la version désirée installée (ROS VM), il est important de noter plusieurs problèmes rencontrés :
- Si le fichier .ova indique une erreur à l'importation, il peut être nécessaire de la reconfigurer avant de la lancer à l'aide des configurations de ce lien (`http://nootrix.com/software/ros-indigo-virtual-machine/`).
- Si le problème persiste, il sera nécessaire de décompresser le fichier ova (WinRar par exemple) et récupérer uniquement le fichier disque sans le fichier configuration)
- Si la machine plante sur l'écran Ubuntu ou freeze à l'utilisation, il est nécessaire de désactiver la carte son.

Au moment de la rédaction de la documentation, la machine virtuelle est fournie avec ROS et l'environnement catkin configuré. Le dossier de travail se trouve à la racine et se nomme catkin_ws. Si pour une raison ou une autre ce dossier n'est pas configuré il suffit de reprendre le tutoriel ROS Universal Robotau paragraphe 3.1.1

A cette étape, la machine virtuelle est démarrée mais la configuration réseau reste à faire. Il est nécessaire d'effectuer 3 opérations pour obtenir le réseau le plus agréable à utiliser possible.

2.2 Configuration réseau

Il existe une contrainte de la part du constructeur de la pince, il n'est pas possible d'utiliser le dhcp. C'est pourquoi il est nécessaire de désactiver le dhcp sur Ubuntu 14.04. Il existe de très nombreux tutoriels sur Internet pour modifier le fichier /etc/network/interfaces et attribuer une ip statique.

L'utilisation d'une machine virtuelle recquiert une configuration des cartes réseaux de la machine et de l'ordinateur. Comme le précise le tutoriel, il faut configurer la carte réseau avec une connexion par pont.

La configuration actuelle de la salle avec l'UR nécessite d'utiliser 2 cartes réseaux virtuelles. Le bras et la pince sont reliés à un routeur local tandis qu'internet vient du WIFI IPB. Il faut donc ouvrir deux connexions par pont pour avoir internet dans la machine virtuelle. il est nécessaire de configurer cette deuxième connexion (fichier /etc/network/interfaces).

Il n'est pas forcément pratique d'utiliser internet dans la machine virtuelle. C'est pourquoi il faut configurer l'os qui héberge la machine. Dans le cas de Windows, il est nécessaire de modifier les paramètres avancés des cartes réseaux. Par défaut, Windows considère Ethernet comme prioritaire sur la liaison Internet si un cable est branché. Il faut définir la carte réseau WIFI comme prioritaire. La procédure officielle : `https://support.microsoft.com/fr-fr/kb/299540`

Il faut attribuer une métrique plus petite à la carte WIFI que à la carte Ethernet.

A cette étape, la machine est complètement configurée et fonctionnelle. L'environnement ROS est prêt et nécessite l'installation de packages pour être utilisé.

3 Installation Package UR10

Cette partie est probablement la plus longue à mettre en place. Il a fallu plusieurs séances pour réussir à faire fonctionner ce package.

3.1 universal_robot

Une fois encore il faut revenir au tutoriel du package. Ce dernier précise l'adresse du repo à copier. Il est important de cloner le dossier dans le répertoire /catkin_ws/src.

Il est maintenant nécessaire d'executer la commande qui permet de sourcer le projet. Cette commande est à executer dans chaque terminal relatif à l'utilisation de ROS.

```
~/catkin_ws source devel/setup.bash
```

Il est temps de compiler l'ensemble et construire le projet. On utilise la commande *catkin_make*. A cet instant, toutes les dépendances sont installées. Il est possible que certains packages Linux soient manquant, il faudra donc les installer avec apt-get puis relancer $catki_make$.

3.2 Configuration Réseau UR10

Avant de pouvoir tester le bon fonctionnement du package, il est nécessaire de définir une adresse IP statique à l'UR10. Dans le menu Configuration du robot/Configuration réseau il est possible de définir une adresse statique.

Dans le cadre de ce travail, on a : 192.168.0.9

Ne pas oublier d'appliquer les paramètres.

Il reste à ping (ping 192.168.0.09) pour vérifier que tout est en place.

Il n'est pas possible de tester, comme le préconise le tutoriel, un programme. La version Polyscope de l'UR10 est 3.3 et le module n'a pas été prévu pour ce firmware. Heureusement, il existe un patch qui vient remplacer certains fichiers et assurer la compatibilité de version.

3.3 ur_modern_driver

Ce module recquiert l'installation de dépendances. Avant de cloner le répertoire (`https://github.com/ThomasTimm/ur_modern_driver`) il faut installer ur_msgs, hardware_interface, and controller_manager grâce au module ros_control(https ://github.com/ros-controls/ros_control)

— Cloner le répertoire de ros_control dans /catkin_ws/src
— Cloner le répertoire de ur_modern_driver dans /catkin_ws/src
— Sourcer si ce n'est pas fait
— Build le projet, /catkin_ws catkin_make

Si tout a bien été suivi, le build a été a son terme. Si ce n'est pas le cas, vérifier qu'il ne manque pas une dépendance.

3.4 Remplacement des fichiers

Pour pouvoir utiliser ur_modern_driver, il est nécessaire de copier : /catkin_ws/src/ur_modern_driver/launch/ dans /catkin_ws/src/universal_robot/ur_bringup. Il est peut être intéressant de conserver une copie des fichiers originaux avant de les écraser.

Si on ouvre les fichiers nouvellement copié, on voit qu'il font directment référence aux fichiers du dossier ur_moder_driver. Ainsi, si l'on souhaite modifier les launcher, il vaut mieux modifier les liens internes des fichiers.

3.5 Test

Il est temps de tester le module et la communication. ROS repose sur le principe de programme indépendant (des noeuds), il est donc logique d'ouvrir un nouveau terminal à chaque création d'un noeud. Ne pas oublier de sourcer le projet dans le terminal.

```
roslaunch ur_bringup ur10_bringup.launch robot_ip:=IP_OF_THE_ROBOT
```

Dans le cadre de ce projet, IP_OF_TH_ROBOT vaut 192.168.0.9.

Il existe d'autres type de launcher dans le dossier bringup. La documentation officielle explique l'intérêt de chacun. Il n'a pas été possible de faire fonctionner correctement Gazebo et MoveIt !

Le programme se lance et indique que le noeud est disponible en proposant un certains nombres de topics et services. Si il est indiqué un problème de modèle qui fait référence à un fichier, il est alors

souhaitable de commenter la ligne du fichier (cette ligne indique la position actuelle du robot qui n'est évidemment pas vraie).

Le fichier test_move.py du dossier universal_robot/ur_driver n'est pas fonctionnel, il faudra privilégier sa version patchée.

Ce programme propose un ensemble de mode pour effectuer une liste de déplacements. Tout repose sur la position actuelle du robot, il est conseillé de le positionner de façon à avoir de la place et de la marge au niveau des angles possibles.

On appelle la bonne version du fichier test avec la commande : (A exécuter dans un nouveau terminal configuré)

```
rosrun ur\_modern\_driver test_move.py
```

Le détail du code montre une trajectoire pré-enregistrée à l'aide d'une classe ROS SimpleActionClient. A l'aide de la fonction send, une liste de points est envoyée au robot. En réalité, cela créé un script URScript qui utilise des combinaisons de servoj et movej/l/p.

Pour utiliser un nouveau script, il ne faut pas oublier de donner les droits en exécution au fichier.

3.6 Remarques

Lorsque le bras est arrêté dans une position proche de la limite de la jointure, alors au redémarrage un message d'erreur apparaît indiquant qu'il faut manuellement déplacer le bras. La solution imparable est de faire attention à éteindre le bras dans une position "moyenne".

Il n'est pas possible de communiquer avec le bras si on se trouve dans un menu de la tablette qui permet un déplacement (onglet déplacement, Se déplacer sur le point de passage). Il y a probablement conflit sur l'entrée qui écoute les commandes.

4 Installation Package WSG 50

4.1 wsg50-ros-pkg

Ce paquet est très simple à installer et n'a posé aucun problème.

— Cloner wsg50-ros-pkg `https://github.com/nalt/wsg50-ros-pkg` dans /catkin_ws/src
— Sourcer si ce n'est pas fait
— Build le projet, /catkin_ws catkin_make

4.2 Configuration Réseau

Le manuel d'utilisation de la pince précise qu'il y a un serveur intégré à la pince qui permet de configurer son adresse IP. Il suffit donc d'accéder à l'adresse par défaut de la pince pour la configurer, à condition d'avoir le DHCP désactivé.

Le rapport précise qu'il existe plusieurs modes de fonctionnement de la pince en effectuant un bilan de avantages et inconvénients. L'utilisation la plus logique est le mode auto-update.

4.3 Test

Pour simplifier le lancement du noeud de la pince, il est possible de modifier directement le launcher.

Dans le fichier /wsg50-ros-pkg/wsg_50_driver/launch/wsg_50_tcp.launch, il est possible de directement définir l'ip de connexion et le mode. Une fois cette étape faite, on lance le noeud :

```
roslaunch wsg_50_driver wsg_50_tcp.launch
```

Le git du projet propose un script pour tester les déplacements de la pince à l'aide du topic goal_position qui est le plus adapté à tous les types d'utilisation.

Malheureusement, pour une raison inconnue la gestion de la force n'est pas bonne et des valeurs aberrantes ressortent parfois en utilisant le topic state.

4.4 Remarques

L'information envoyée à la pince est brute. Donc si l'information envoyée est d'un mauvais type ou qu'il est envoyé une information qui va forcer sur les limites de la pince alors la pince se bloque. La seule solution est un redémarrage physique. Une mauvaise commande doit probablement être responsable du crash de la pince.

5 Installation Cinématique Python

Afin de pouvoir utiliser la cinématique de l'UR depuis un script python, il faut installer un module supplémentaire.

5.1 ur_kin_py

Cloner ur_kin_py `https://github.com/gt-ros-pkg/ur_kin_py` dans /catkin_ws/src

Cloner Boost.NumPy `https://github.com/gt-ros-pkg/Boost.NumPy` dans /catkin_ws/src

Sourcer si ce n'est pas fait

Build le projet, /catkin_ws catkin_make

5.2 Test

Le git du projet propose un script de test de la cinématique

Il ressort de ce test que très rarement la cinématique inverse est erronée mais sans réussir à comprendre quelle est la raison.

Notes

La trajectoire en arc de cercle n'est pas fonctionnelle malgré de nombreuses tentatives et essais infructueux.

Tous les scripts peuvent prendre en argument :

1. debug : affiche un certain nombre d'informations sur le déroulement du script
2. debug full : affiche des informations complémentaires.