Rapport de stage assistant ingénieur
Sensor Data Analysis on the Kuka Youbot

IRLab

Juliette FAURY ROB25

Remerciements

Je tiens à remercier mon maître de stage, Arnoud Visser, de m'avoir accueilli au sein de son laboratoire, fourni son aide et des pistes durant ces trois mois et demi, et surtout de m'avoir dit, à la fin de ce stage aux résultats négatifs, que le plus important était d'avoir acquis des connaissances.

Je remercie également le service électronique de l'Université d'Amsterdam pour nous avoir créé et fourni un nouveau connecteur à partir de deux en seulement une trentaine de minutes.

Résumé

Le KUKA youBot est un manipulateur mobile, principalement développé pour l'éducation et la recherche. Il se compose de deux parties principales :

— La plateforme mobile omni-directionnelle KUKA youBot est constituée du châssis du robot, de quatre roues Mecanum, de moteurs, d'une alimentation électrique et d'une carte PC embarquée. Les utilisateurs peuvent soit exécuter des programmes sur cette carte, soit la contrôler depuis un ordinateur à distance.

— Le bras KUKA youBot dispose de cinq degrés de liberté (DOF) et d'une pince à deux doigts. Lorsqu'il est connecté à la plateforme mobile, le bras peut être contrôlé par le PC embarqué. Alternativement, le bras peut être contrôlé sans la plateforme mobile en utilisant un PC personnel connecté via un câble Ethernet.

Des capteurs supplémentaires peuvent être montés sur le robot. Le but de ce stage était de mettre en œuvre une méthode SLAM (localisation et cartographie simultanées) avec un Lidar Velodyne. Mais au cours des premières semaines, ROS Hydro a été accidentellement supprimé du robot. Le reste du stage a donc consisté à tenter de réinstaller correctement ROS (il n'était téléchargable de la manière dont cela avait été fait il y quelques années, car il était devenu obsolète). Le logiciel pu être partiellement restauré à la fin du stage.

Abstract

The KUKA youBot is a mobile manipulator that was primarily developed for education and research. It's consists of two main parts:

— The KUKA youBot omni-directional mobile platform consists of the robot chassis, four mecanum wheels, motors, power and an onboard PC board. Users can either run programs on this board, or control it from a remote computer.

— The KUKA youBot arm has five degrees of freedoms (DOF) and a two-finger gripper. If connected to the mobile platform, the arm can be controlled by the onboard PC. Alternatively, the arm can be controlled without the mobile platform by using an own PC connected via Ethernet cable.

Additional sensors can be mounted on the robot. The purpose of my internship was to implement a SLAM (Simultaneous location and mapping) method with a Velodyne Lidar. During the first weeks, ROS Hydro was accidentally removed from the robot. The rest of the internship consisted in trying to reinstall correctly ROS (the sofware wasn't available anymore in the usual download methods, because it had become too outdated). I suceeded in partially restoring the software, but some parts of it were still not functioning properly.

Table des matières

1 Présentation du contexte de stage **1**

2 Problématisation **1**

3 Description de l'activité et des résultats du stage **2**

3.1 Prise en main du robot et premiers tests 2

3.2 Tests des drivers avec le télémètre laser Hokuyo 4

3.3 Préparation du Velodyne 5

3.4 Tentatives de refaire fonctionner le `catkin_make` après une mauvaise commande et effacement de ROS 6

3.5 Tentatives de réinstaller ROS depuis les sources et problèmes de packages manquants 7

3.6 Procédure d'installation des packages récupérables 8

3.7 Récupération d'une partie de ROS via la copie des fichiers ROS d'un autre ordinateur 8

3.8 Problème de compatibilité processeur 9

3.9 Solution de l'erreur -lpthread 9

3.10 Test de la version Ubuntu 14.04 10

4 Conclusion **11**

Bibliographie **12**

Liste des Figures **13**

Annexe 1: Fiche d'évaluation **13**

1 Présentation du contexte de stage

Le Intelligent Robotics Laboratory (IRLab) est un laboratoire de l'Université d'Amsterdam ayant pour objectif la collaboration entre chercheurs et élèves, dans l'enseignement et la recherche en robotique [1]. Le but de ce laboratoire est de fournir aux étudiants une expérience pratique dans l'utilisation de systèmes d'intelligences artificielles. Pour cela, étudiants et enseignants ont accès à de nombreux matériels et ressources, notamment des robots et drones. L'activité principale du laboratoire est la préparation et la participation au concours annuel RoboCup avec une équipe de robots NAO.

Le but de ce stage était l'implémentation d'un LIDAR récemment acquis par le laboratoire sur un bras robot KUBA Youbot. Ce travail a été effectué en collaboration avec Basile MOLLARD, étudiant en robotique à l'ENSTA Bretagne, également en stage au sein du IRLab. L'objectif sous-jacent de ce stage était de nous permettre d'acquérir de nouvelles compétences grâce à une expérimentation concrète sur un robot, dans la lignée de l'aspect formation du laboratoire.

2 Problématisation

Initialement, ce stage devait permettre l'implémentation d'un nouveau radar et d'une nouvelle méthode de localisation sur un robot. Cependant, suite à un incident, le sujet a été réorienté vers la réparation de la partie logiciel du robot. Pour cela, il fut nécessaire de pouvoir identifier les méthodes peu efficaces sur lesquelles il fallait ne pas s'obstiner. En raison de la nature du laboratoire, orienté vers la formation, les enjeux de ces travaux n'étaient pas trop importants.

3 Description de l'activité et des résultats du stage

3.1 Prise en main du robot et premiers tests

Lors de ce stage, le robot utilisé était un YouBot de l'entreprise KUKA, spécialisée dans les solutions d'automatisations intelligentes. Le YouBot fut initialement développé pour la recherche et l'enseignement [2]. Il est composé d'une plateforme mobile omni-directionelle ainsi que d'un bras à cinq degrés de liberté avec une pince préhensile (voir figure 1). Le robot dans son ensemble est entièrement contrôlable avec un ordinateur de bord intégré dans la plateforme. La totalité du travail ayant été effectuée sur la plateforme, le bras fut seulement utilisé pour vérifier la connexion entre les deux ensembles.

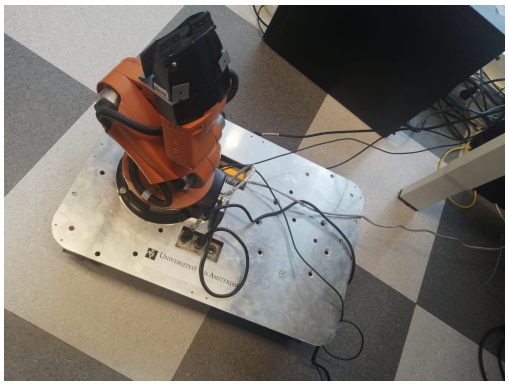

FIGURE 1 – Le robot Kuka YouBot

Il a tout d'abord fallu réussir à allumer et connecter les différentes parties du robot. Sur le PC embarqué doivent être branchés un cable ethernet, un clavier et une souris (il y a un port VGA et des ports USB supplémentaires à cet effet sur le flanc du robot, visibles sur la figure 4). Pour connecter le bras robotisé, ce dernier doit être également relié au PC embarqué sur sa prise EtherCAT, ainsi qu'à la partie alimentation électrique du robot (figure 5). Cette partie est reliée à une prise secteur, ou à la batterie interne du robot en fonction des besoins.

L'allumage se fait via un bouton sur le panneau du PC embarqué (le bouton rond noir sur la figure 2). Il est possible d'allumer uniquement le PC embarqué, uniquement les moteurs, ou d'allumer le tout. Pour pouvoir déplacer la plateforme ou le bras, il faut que les moteurs soient allumés. Pour pouvoir utiliser le bras, il faut en plus l'allumer spécifiquement via un un bouton situé sur sa base (illustré par la figure 2).

Le système d'exploitation du PC embarqué du robot est Ubuntu 12.04 (Precise Pangolin), une version en fin de vie depuis avril 2017 [3], maintenue ici en raison des compatibilités avec les différents drivers de la plateforme et du bras. La version de ROS qui sert à communiquer avec les drivers est ROS Hydro Medusa.

Le premier test a été de lancer le programme de démonstration `hello_world_demo`, qui translate le robot d'environ un mètre vers la droite puis d'un autre mètre vers la gauche avant de replier le bras sur lui même. Ensuite, les programmes permettant de déplacer la plateforme via le clavier et de donner des commandes angulaires aux

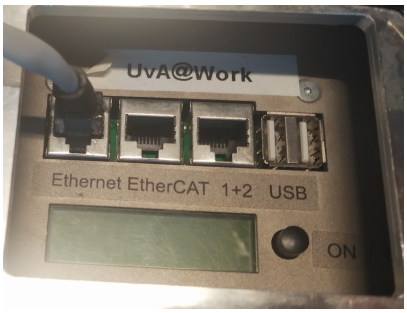

FIGURE 2 – PC embarqué

FIGURE 3 – Base du bras du KUKA

FIGURE 4 – Flanc du robot Kuka

FIGURE 5 – ports d'alimentation électriques

actionneurs du bras, avec ROS, ont été testés. Pour cela, il a fallu au préalable faire la configuration réseau entre le PC embarqué et le bras afin de pouvoir lancer les différents drivers contrôlant les moteurs sans erreurs.

Il a fallu par la suite recommencer plusieurs fois la configuration réseau car l'adresse IP du port Ethernet changeait parfois au rallumage du PC embarqué. Elle consistait à chercher l'adresse IP du port `eth0` (correspondant à la liaison bras-pc embarqué) en tapant dans un terminal:

```
$ ip addr show eth0
```

Une ligne de ce type apparaissait

```
inet 10.42.0.1/24 brd 10.42.0.255 scope global eth0
```

Et ensuite il suffisait d'ouvrir le bashr

```
$ nano ~/.bashrc
```

d'ajouter (11311 est le port ROS par defaut)

```
export ROS_IP=10.42.0.1
export ROS_MASTER_URI=http://10.42.0.1:11311
```

puis de sauvegarder et de sourcer le bashrc. Après avoir réglé les problèmes de réseaux et compilé certains fichiers manquants, la plateforme et le bras étaient contrôlables au

clavier, montrant que les drivers liés aux moteurs étaient opérationnels. La prochaine étape était de tester tout ce qui était lié à la partie capteur et navigation.

3.2 Tests des drivers avec le télémètre laser Hokuyo

Le Kuka robot est initialement conçu pour pouvoir mettre en place une navigation avec le télémètre laser Hokuyo URG-04lx [2], un capteur laser pour la numérisation de zones (visible sur la figure 6). Sa zone de balayage est de 240° avec un rayon maximal de 4 m. Le principe de mesure de distance de ce capteur est basé sur le calcul de la différence de phase, qui permet d'obtenir une mesure stable avec une influence minimale de la couleur et de la réflectance de l'objet. Les données de balayage laser fournies sont sous forme de portée et d'angle de relèvement [4].

Le but de ce stage était d'implémenter une méthode de navigation type SLAM (*simultaneous location and mapping*) sur le robot Kuka avec un autre capteur, un Lidar Velodyne. Or, en lisant la documentation du robot, il est apparu qu'il existait déjà une pile de navigation ROS implémentée, mais adaptée au capteur Hokuyo. Il a alors été décidé de tester d'abord les nœuds ROS avec le Hokuyo (que possédait le laboratoire) pour la prise en main, avant une adaptation potentielle au Velodyne.

FIGURE 6 – Le capteur connecté sur le flanc du robot

Après avoir connecté le télémètre, l'application `vmon` a permis une première visualisation des données émises par le capteur (voir figure 7). Lors de la pause d'un obstacle devant le capteurs, des changements de données cohérents étaient observables, assurant un télémètre fonctionnel.

Pour les nœuds ROS autour du Hokuyo, il y avait des instructions dans le guide utilisateur du Kuka. Les instructions concernant le SLAM et les nœuds associés étaient lacunaires car provenant du wiki de l'entreprise Kuka, dont certaines parties n'étaient plus accessibles car fermées avec le temps (et non sauvegardées sur des sites de récupération d'anciens sites web tel que Wayback machine).

Il a donc juste été tenté de lancer un maximum de nœuds qui étaient liés à la navigation. Le nœud lié au télémètre laser Hokuyo devait être reconfiguré pour pouvoir être lancé correctement. Il fallait juste modifier le port dans le fichier de launch. Par contre, il a fallu entièrement créer `youbot_state_publisher.launch` (après avoir récupéré le package correspondant sur git), car lors du lancement d'un des nœuds, il y avait l'erreur:

```
    the 'state\_publisher' executable is depreciated. Please use 'ro-
bot\_state\_publisher' instead
```

Une fois un certains nombre de nœuds ROS liés à la navigation actifs, `Rviz` a été utilisé pour pouvoir créer une carte grossière du robot dans l'environnement capté par le télémètre, visible sur la figure 8. Grace à un tutoriel `rviz` [5] permettant de connecter un nœud lié au moteur, le robot bougeait lorsque l'on cliquait sur la carte de l'environnement (à terme cela aurait permis déplacer le robot dans une carte crée via le SLAM par exemple). Il a également été possible de créer une carte montrant les liens entre les différents nœuds lors des test des drivers (figure 9).

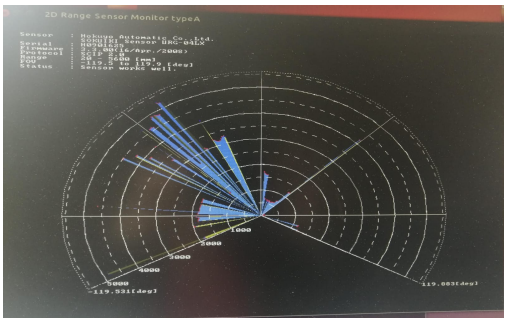

FIGURE 7 – Illustration d'une application de `vmon` au télémètre laser Hokuyo

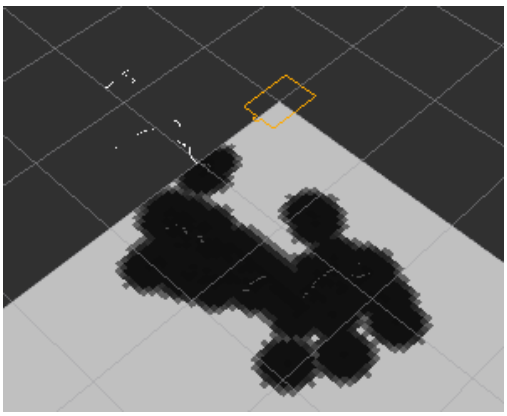

FIGURE 8 – Capture d'écran de `Rviz` : en orange le robot Kuka, en noir les obstacles détectés par le capteur Hokuyo

Le Velodyne offre la possibilité de la navigation 3D (il existe des nœuds ROS rendant faisable cette navigation). Un de nos objectifs futurs était de tester cette navigation en trois dimensions en ajoutant des nouveaux nœuds au robot Kuka.

3.3 Préparation du Velodyne

Le lidar à 360° de Velodyne, qui devait être utilisé à la place du capteur, nécessitait une prise secteur pour fonctionner. Or le lidar étant placé sur la plateforme du Kuka

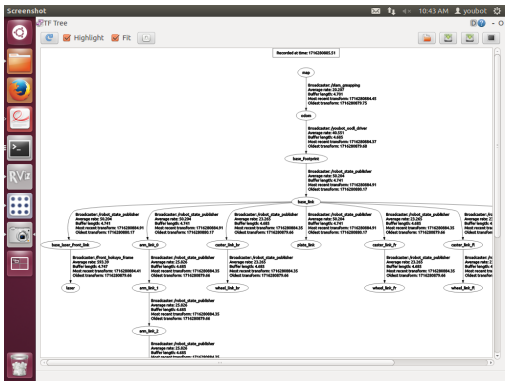

FIGURE 9 – Diagramme des nœuds lors des tests des driveurs

lors de la navigation, il fallait le rendre indépendant d'une prise secteur. La première idée était de le brancher directement sur le robot, mais il fallait fournir 12V et 2,1A pour 25W maximum. Les ports USB du PC embarqué ne pouvaient fournir que 5V, tandis que l'alimentation intégrée au robot pour le bras fournissait 24V. Notre encadrant de stage nous a alors fourni une batterie externe correspondant aux caractéristiques électroniques de notre lidar. Mais il restait un problème : la connectique entre le câble permettant d'alimenter le Velodyne et de le connecter à l'ordinateur (qui avait un embout non standard) et le câble de la batterie externe. Heureusement, le service électronique de l'université a été capable de nous fabriquer un intermédiaire fonctionnel, visible sur la figure 10, permettant finalement au lidar de pouvoir être sur la plateforme mobile sans contraintes.

FIGURE 10 – La batterie externe, le nouveau connecteur et le Velodyne

3.4 Tentatives de refaire fonctionner le `catkin_make` après une mauvaise commande et effacement de ROS

Lorsque nous avons essayé de connecter et de configurer le Velodyne, nous avons effectué une commande `boost purge`, qui effaça malencontreusement de nombreuses liaisons entre les différentes librairies et `catkin`, notre outil de compilation. Dès lors, les tentatives d'utiliser `catkin_make` pour compiler notre fichier de travail ne se soldèrent

qu'en une quantité importante d'erreurs, majoritairement liées au fait que Catkin ne trouvait plus les packages ROS.

En réinstallant via `git clone` et `cmake` les packages affichés dans les erreurs (par exemple `ros_comm`, `common_msgs`, `message_generation`, `message_runtime`, `std_msgs`), une partie de ces dernières ont disparues. Il a fallut également parfois ajouter des chemins de fichiers dans le `catkinConfig.cmake` (pour une dizaine de noms de fichiers). Pour certains fichiers (comme `std_msgs`), il était nécessaire de copier-coller des fichiers de partages de librairies là où le compilateur Catkin allait les chercher (en se guidant d'après les erreurs).

En essayant de réinstaller un des différent packages de ROS, nous avons effectué la commande

```
catkin_make --only-pkg-with-deps rosgraph_msgs
```

Or, nous ignorions qu'après avoir réaliser cette commande, que seuls les packages indiqués dans cette dernière seront recompilés lors des futurs `catkin_make` (sans précisions de packages). Il a donc été considéré à tort que tous les packages du workspace catkin avait été réparés et étaient compilables. Sauf que, en voulant réessayer le programme permettant de contrôler le robot en clavier, qui marchait avant l'utilisation du `boost purge`, il est devenu visible que la commande `roscore` permettant de lancer ROS Hydro ne fonctionnait plus. L'erreur suivante s'affiche

```
    ImportError: No module named roslib.packages
```

Nous avons alors suivi une commande trouvée sur Internet:

```
sudo aptitude reinstall ros-hydro-ros
```

et ROS a été effacé, et non réinstallé car plus disponible via `apt install` à cause de l'âge de ROS Hydro. Notre maître de stage nous a conseillé ultérieurement de toujours effectuer un "dry run" dans ce genre de situations pour éviter les accidents.

3.5 Tentatives de réinstaller ROS depuis les sources et problèmes de packages manquants

Les tentatives de réinstaller ROS avec `apt` ne fonctionnant pas, il a été essayé de rajouter des dépôts APT et d'utiliser des outils facilitant l'installation de ROS comme `wstool` et `rosinstall_generator`, mais les packages restaient introuvables par l'ordinateur. Un essai avec une librairie nommée Poco a été effectué, mais une des dépendance était incompatible avec le processeur du PC embarqué. Poco a réussi à être installé d'une autre façon mais lors de son utilisation il y avait une erreur concernant une librairie "-lpthread" qui n'a pas pu être enlevée. Le git du workspace catkin de ROS a alors été cloné, pour tenter de compiler depuis les sources. Certaines erreurs de compilation étaient similaires à celles rencontrées précédemment, donc aisément solvables. Mais il y a eu une erreur qui a causé beaucoup de soucis tout le long du reste du stage, la même qui avait complètement bloqué l'utilisation de Poco:

```
CMake Error at /home/youbot/ros_hydro_ws/devel_isolated/class_loader/
    ↪ share/class_loader/cmake/class_loaderConfig.cmake:141 (message):
  Project 'class_loader' tried to find library '-lpthread'. The library
     ↪ is
  neither a target nor built/installed properly.
```

Le Project inscrit dans l'erreur variait en fonction du package compilé, mais la forme globale de l'erreur était toujours la même que ci dessus.

Après avoir tenté de compiler le package individuellement, il a été essayé de rajouter des lignes dans le fichier CMake du package demandant la librairie threads, de changer la version de CMake lors de la compilation, de changer le fichier de configuration du package pour enlever la nécessité de cette librairie, et de rajouter une ligne dans le CMake enlevant -lpthread des librairies du package, mais rien ne voulait faire disparaître cette erreur. Finalement, il est apparu dans le CMake que la librairie Boost était utilisée pour chercher -lpthread, donc la demande de Boost inscrite dans le CMake a été remplacée par la demande de la librairie Threads. Cela a permis de compiler 41 packages de ROS sans erreurs, avant que des erreurs de packages manquants apparaissent.

Il a fallu alors trouver les librairies manquantes, et les installer soit via apt, soit via les sources. Mais il commençait à y avoir un certain embourbement dans de nombreuses erreurs, qui nécessitait de compiler et de debuger toujours plus de packages.

3.6 Procédure d'installation des packages récupérables

La plupart des petites librairies extérieures s'installaient correctement avec une compilation g++, les principaux problèmes viennent des interdépendances des packages catkin (non compilable avec g++), car la commande `catkin_make` était inutilisable, et la commande `catkin_make_isolated` tente de compiler tous les packages du workspace les uns après les autres. Après de nombreux tâtonnements, une manière de compiler et d'installer un package individuellement a été trouvé. Ici un exemple avec `cpp_common`:

```
$./src/catkin/bin/catkin_make_isolated --only-pkg-with-deps cpp_common --
    ↪ install -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="-std=c++98"
```

(certains packages nécessitent plûtot la balise `CMAKE_CXX_FLAGS=""`)

Cela a amélioré grandement la récupération des packages, car une fois le package installé "correctement", le nombre d'erreurs liées à des problématiques de mauvaises liaisons entre les packages a grandement diminué.

3.7 Récupération d'une partie de ROS via la copie des fichiers ROS d'un autre ordinateur

Notre encadrant, nous voyant en difficulté depuis plusieurs jours sur nos tentatives de reconstruction de ROS depuis les sources, sans grands espoirs de succès, nous fournit un autre ordinateur avec ROS Hydro. Le but était de récupérer le ROS de ce deuxième

ordinateur pour le mettre sur notre robot. Avant cela nous avions cherché la clef du software du Kuka Youbot, mais sur les deux trouvées dans le laboratoire, aucune n'était complètes. Pour récupérer une nouvelle clé, il aurait fallu faire une demande via le laboratoire au constructeur, sans garantie de réponse. Nous avons, via une clé USB, copié-collé le fichier `/opt/ros/hydro` (qui avait disparu du robot lors de la commande `ros reinstall`) de l'ordinateur fourni via notre maître de stage via le PC embarqué du Kuka.

Une fois les fichiers ROS Hydro de l'ordinateur fourni récupérés, il fallut les mettre au bon endroit sur notre PC embarqué, à l'endroit où le précédent package ROS était installé. Il a fallu ensuite ajouter des permissions pour pouvoir sourcer ce nouveau ROS. Pour tester cette installation, la commande `roscore`, démarrant le système ROS sur la machine, a été lancée. L'erreur produite avait changé : il manquait maintenant un package python, nommé `netiface`. Il a été un peu délicat à retrouver, car seule l'utilisation d'une version miroir obtenue sur le site d'une université a réussi à faire disparaître l'erreur. Il y avait encore différentes erreurs lors du lancement de roscore. Il s'est avéré qu'un module important de ROS, rosout, n'était pas construit et que le module rosmake permettant de compiler les packages ROS n'était pas utilisable non plus. Rosmake avait des problèmes de permissions, et en cherchant sur les forums, il été souvent indiqué lors du débugage d'erreurs d'ajouter des précisions sur la permissions avec la commande `ls -lah`. Cette commande dans le fichier ROS transféré d'un ordinateur à un autre a montré une absence de certaines permissions. Après avoir rajouté les permissions nécessaires, l'erreur de permission a disparu mais des packages restent manquants.

3.8 Problème de compatibilité processeur

Au cours des divers debugages et compilations qui ont été effectuées ensuite, il y a eu une erreur remarquable, qui n'était encore jamais apparue, lors de certaines compilation:

```
/opt/ros/hydro/lib/librostime.so: could not read symbols: File in wrong
    ↪ format
```

Après quelques essais infructueux, des recherches ont révélé que les fichiers .so étaient des fichiers contenant du code machine. Or ces fichiers .so provenaient d'un autre ordinateur, le code machine n'était donc pas le même. Pour résoudre ce problème, un peu de triche a été utilisée : on réinstallait la librairie dans le workspace avec `catkin_make_isolated`, puis on déplaçait le fichier en .so du folder `install_isolated` du workspace vers l'emplacement du fichier .so en mauvais langage machine pour remplacer ce dernier.

Il y a eu après cela une phase où les erreurs consistaient principalement à trouver, compiler et/ou installer des librairies, des packages et des dépendances, ainsi qu'à régler des problèmes de permissions.

3.9 Solution de l'erreur -lpthread

Une des erreurs qui bloquait toujours la compilation de certains packages (notamment ceux des packages contenant les différents messages ros) était l'échec de la recherche de

la librairie -lpthread (un exemple d'erreur de ce type est présenté au 3.5). Nous avons réussi à identifier que l'erreur venait des CmakeList.txt, plus précisément des lignes de formes: "find_package(catkin REQUIRED COMPONENTS ". Des multiples tentatives de faire fonctionner cette ligne telle quelle ont été effectuées, au final il a fallu passer par un gestionnaire de packages nommé PkgConfig. Voici un exemple de modification de CmakeList.txt modifié. Les lignes commentées sont celles qui ont été retirées et celles entre les triples hashtag sont celles ajoutées :

```
cmake_minimum_required(VERSION 2.8.3)
project(rosout)
###
set(ENV{PKG_CONFIG_PATH} "/opt/ros/hydro/lib/pkgconfig")
find_package(PkgConfig REQUIRED)
pkg_check_modules(ROSCPP REQUIRED roscpp)
include_directories(${ROSCPP_INCLUDE_DIRS})
link_directories(${ROSCPP_LIBRARY_DIRS})
add_definitions(${ROSCPP_CFLAGS_OTHER})
###
#find_package(catkin REQUIRED COMPONENTS roscpp)

catkin_package()
include_directories(${catkin_INCLUDE_DIRS})
add_executable(rosout rosout.cpp)
#target_link_libraries(rosout ${catkin_LIBRARIES})
###
target_link_libraries(rosout ${ROSCPP_LIBRARIES})
###
install(TARGETS rosout
  RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
```

Il a fallu répéter cette opération pour toute une liste de packages, en sachant que certains d'entre eux avaient d'autres erreurs de compilation à corriger.

3.10 Test de la version Ubuntu 14.04

Il a été tenté comme solution alternative d'upgrader la version d'Ubuntu du PC embarqué pour pouvoir utiliser ROS Indigo. Pour cela une clé bootable Ubuntu 14.04 a été réalisée. Mais Ros Indigo n'a pas pu être installée en utilisant la version clé bootable car la version de libbost nécessaire n'était plus installable. Après avoir passé quelques jours dessus, cette solution a été abandonnée car n'ayant pas de garantie de résolution aux problèmes de librairies.

4 Conclusion

Ce stage fut l'occasion de beaucoup pratiquer la manipulation, la fabrication, la configuration, l'installation et le debugage de librairies, ainsi que de travailler avec de nombreuses commandes terminal Linux et l'organisation des fichiers sur une machine. Néanmoins, une obstination et un entêtement trop fort à vouloir persévérer sur la voie de la réparation manuelle des fichiers ROS supprimés ont été néfastes sur le projet en empêchant la recherches plus intense de solutions alternatives qui aurait pu permettre un avancement sur la partie navigation du sujet. A la lumière des connaissances acquises depuis le stage, il aurait été pertinent d'au moins essayer de créer une image Docker avec des fichiers partagés. Nous avons également appris durement que sauvegarder le software d'un robot avant de commencer à le modifier est une étape essentielle.

Références

[1] U. van Amsterdam, "Education labs," https://ivi.uva.nl/education/education-labs/education-labs.html, consulté le: 10 septembre 2024. 1

[2] Locomotec, *KUKA youBot User Manual*, Dec. 2012, version 1.02, ©Locomotec. 2, 4

[3] W. ubuntu fr, "Ubuntu 12.04 lts (« the precise pangolin »)," https://doc.ubuntu-fr.org/precise, consulté le: 29 septembre 2024. 2

[4] H. Suresh and N. Surathkal, "Slam and navigation using lidar for a kuka youbot," Indian Institute of Technology Madras, Project Report, Jul. 2016. 4

[5] Ros.org, "Using rviz with the Navigation Stack," `https://wiki.ros.org/navigation/Tutorials/Using rviz with the Navigation Stack.html`, consulté le: 16 septembre 2024. 5

Table des figures

1 Le robot Kuka YouBot 2
2 PC embarqué 3
3 Base du bras du KUKA 3
4 Flanc du robot Kuka 3
5 ports d'alimentation électriques 3
6 Le capteur connecté sur le flanc du robot 4
7 Illustration d'une application de `vmon` au télémètre laser Hokuyo 5
8 Capture d'écran de `Rviz` : en orange le robot Kuka, en noir les obstacles détectés par le capteur Hokuyo 5
9 Diagramme des nœuds lors des tests des driveurs 6
10 La batterie externe, le nouveau connecteur et le Velodyne 6

ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à : *At the end of the internship, please return this report via mail or email to:*

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Universiteit van Amsterdam

Adresse / *Address* Science Park 900

Tél / *Phone (including country and area code)* +31653697548

Nom du superviseur / *Name of internship supervisor*
Arnoud VISSER
Fonction / *Function* Senior Lecturer in Robotics

Adresse e-mail / *E-mail address* a.visser@uva.nl

Nom du stagiaire accueilli / *Name of intern* Juliette FAURY

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)** *Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? A B **C** D E F
  Was the initial contract carried out to your satisfaction?

- Manquait-il au stagiaire des connaissances ? ■ oui/*yes* ☐ non/*no*
  Was the intern lacking skills?

Si oui, lesquelles ? / *If so, which skills?* The student learned a lot of working on a robot purely via the command-line. In addition, the student learned a lot on installation dependencies, building libraries from source code and debugging installation procedures.

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*

A B **C** D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*
Juliette is a nice person, but introvert and a bit shy, so not a natural team player. In addition, she felt insecure about her level of English, which withheld her from spontanious joining the conversations in the lab.

COMPORTEMENT AU TRAVAIL / BEHAVIOUR TOWARDS WORK

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

A(B)C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* Juliette works very consistently on the tasks given to her. She should only be more assertive to give feedback when a task given by the management is impossible.

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A(B)C D E F (Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A **B** C D E F

(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

She worked very autonomously and tried different ways to solve the problems encountered.

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B(C)D E F *Was the intern open to listening and expressing himself /herself?*

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* *See team spirit*

OPINION GLOBALE / *OVERALL ASSESSMENT*

✞ La valeur technique du stagiaire était : A(B)C D E F

Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

✞ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ■ oui/*yes* ☐ non/*no*

Fait à _____ , le _____ In Amsterdam , on __ September 3, 2024___

Signature Entreprise _____ Signature stagiaire

Company stamp _____ *Intern's signature*

Instituut voor Informatica
Universiteit van Amsterdam
Science Park 904
1098 XH Amsterdam
T +31 20-525 7460
www.science.uva.nl

Merci pour votre coopération
We thank you very much for your cooperation