Laboratoire sur les robots autonomes simulés

IFT608 / IFT702
Planification en intelligence artificielle
Hiver 2023
Professeur : Froduald Kabanza
Assistant : Jordan Félicien Masakuna

Pour maîtriser les algorithmes de planifications, il est important de les appliquer. Un des domaines d'application est la robotique. Ce document décrit trois laboratoires qui vous permettront de vous familiariser avec des outils de contrôler des robots autonomes (ROS2), les simuler (Gazebo), les visualiser dans leur environnement (Rviz) et les rendre autonomes en les équipant des capacités de planifier des trajectoires (OMPL) et des des tâches (PlanSys2).

Pour ce laboratoire, si vous n'avez pas réussi à installer les outils par vous-même, vous devez utiliser la machine virtuelle qui vous a été fournie. Cette machine, fonctionnant sous Ubuntu 22.04, possède tous les logiciels nécessaires pour mener à terme ce projet : ROS2, Gazebo, RViz et PlanSys2.

Pour les points techniques, référez-vous aux manuels et forums de ROS. Vous pouvez aussi l'assistant du cours. Elle pourra vous répondre dans une certaine mesure.

1 Familiarisation de base

Les six laboratoires suivants ont pour but de vous familiariser avec les outils utilisés (ROS, Gazebo, RViz, MoveIt, Nav2 et PlanSys). Gazebo et RViz sont des outils de visualisation. Gazebo indiquera le détail sur l'environnement tandis que RViz permettra de visualiser le modèle de robot simulé, d'enregistrer les informations des capteurs du robot et de relire les informations des capteurs enregistrées.

1.1 Machine virtuelle

Ceux qui vont installer ces différents outils directement sur leurs machines (plutôt que sur une machine virtuelle), et ceux qui vont installer une autre version de machine virtuelle, n'auront pas besoin de considérer cette étape. Sinon, télécharger la machine virtuelle

https://usherbrooke-my.sharepoint.com/:u:/g/personal/masj2413_usherbrooke_ca/ETd0L_o_Q4hPoW1FTApLajEB9Rx3lGU-rgdbdZ_AATEf5g?e=f5IOlL

Le mot de passe est : @@Ker10

Ce tutoriel fait référence aux vidéos qui sont dans la machine virtuelle dans l'emplacement suivant : **~/Videos/Screencasts.**

1.2 Installation du système d'exploitation robotique (ROS) sur Ubuntu

Le système d'exploitation robotique (ROS) est un ensemble de bibliothèques logicielles et d'outils qui permettent à créer des applications robotiques. La version ROS utilisée est Humble.

Suivez la procédure d'installation du ROS telle que reprise dans la vidéo ***ROS2_installation.webm****. Vérifiez le fonctionnement du ROS comme indiqué vers la fin de la vidéo. L'exercice va constituer à lancer un subscriber.*

Ressource: https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

1.2.1 Tâches à exécuter

1.2.1.1 Installation

Sur votre terminal, executez les commandes suivantes:

1. Activez l'encodage UTF

   locale # check for UTF-8
   sudo apt update && sudo apt install locales
   sudo locale-gen en_US en_US.UTF-8
   sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
   export LANG=en_US.UTF-8
   locale # verify settings

2. Activez la lecture du répertoire Ubuntu

   sudo apt install software-properties-common
   sudo add-apt-repository universe

3. Ajouter la clé GPG de ROS2

   sudo apt update && sudo apt install curl
   sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg

4. Ajouter le répertoire Ubuntu a votre source

   echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null

5. Mettez à jour votre système

   sudo apt update

sudo apt upgrade

6. Installez ROS Humble (Humble est la dernière version de ROS)

   sudo apt install ros-humble-desktop

7. Installez les librairies de communication, messages et terminal

   sudo apt install ros-humble-ros-base

8. Installer les outils de développement sudo apt install ros-dev-tools

   sudo apt install ros-dev-tools

9. Ajoutez le dossier d'installation ROS a votre fichier ~/.bashrc

   echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc

1.2.1.2 Vérification

1. Lancez un nouvel terminal et executez la commande suivante:

   ros2 run demo_nodes_cpp talker

2. Lancez un autreterminal et executez la commande suivante:

   ros2 run demo_nodes_py listener

Le *listener* devrait recevoir les messages envoyés par le *talker* (Figures 1 et 2).

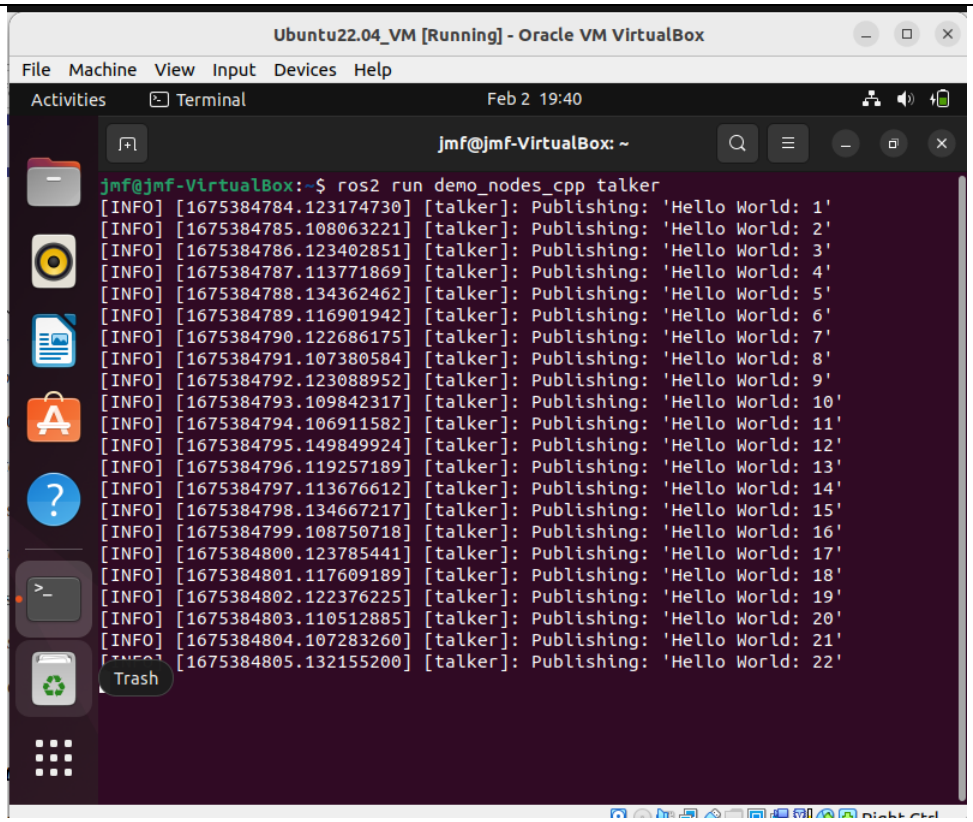

Figure 1: Talker

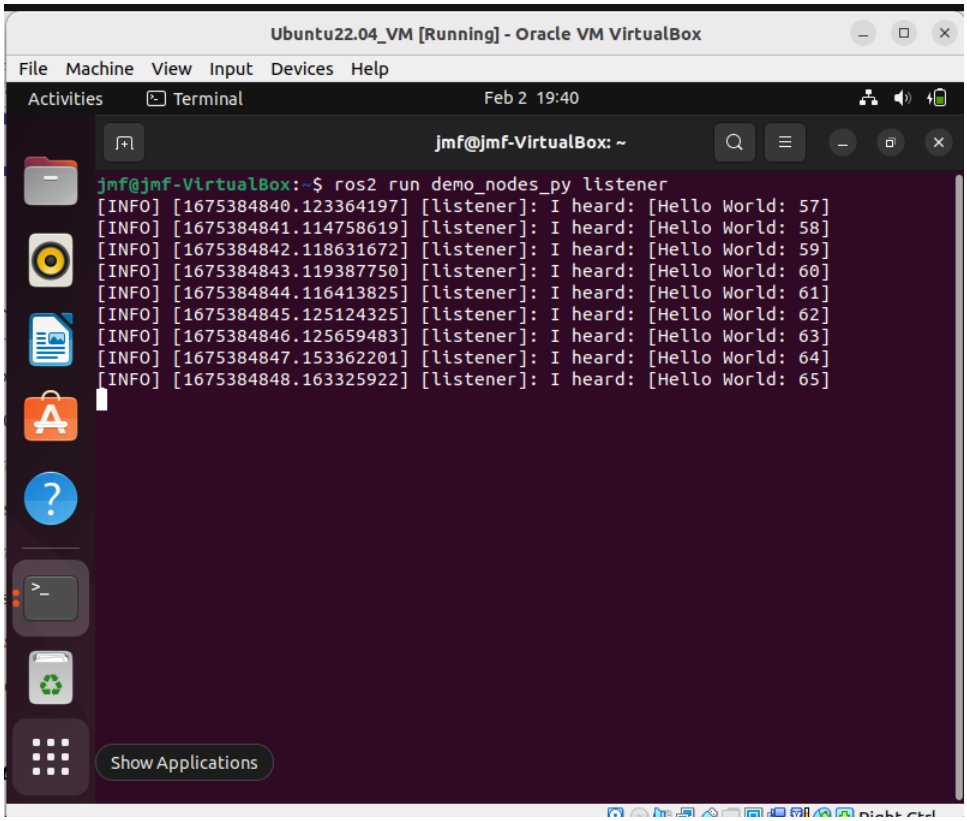

Figure 2: Listener

1.3 Installation et utilisation du package MoveIt

MoveIt est la plate-forme de manipulation robotique pour ROS et intègre les dernières avancées en matière de motion planning, manipulation, perception 3D, cinématique, contrôle et navigation (https://moveit.picknik.ai/humble/index.html).

Suivez la procédure d'installation du package MoveIt telle que reprise dans la vidéo ***Moveit_installation.webm****. Vérifiez le fonctionnement du MoveIt comme indiqué dans les vidéos suivantes:* ***Moveit_Tutorial.webm*** *et* ***Moveit_Tutorial2.webm****. Le premier exercice va constituer à lancer un robot Panda Arm afin de faire des manipulations des trajectoires. Un deuxième tutorial va constituer à créer un package.*

Ressource: https://moveit.picknik.ai/humble/doc/tutorials/getting_started/getting_started.html

1.3.1 Tâches à exécuter

1.3.1.1 Installation

1. Installez la commande colcon et ses dépendances

   sudo apt install python3-colcon-common-extensions
   sudo apt install python3-colcon-mixin
   colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-repository/master/index.yaml
   colcon mixin update default

2. Créez un dossier nommé *ros2_ws/src* dans votre home

   mkdir -p ~/ros2_ws/src

3. Clonez le package moveit

   cd ~/ros2_ws/src
   git clone https://github.com/ros-planning/moveit2_tutorials -b humble --depth 1

4. Téléchargez les tutorials du package Moveit

   vcs import < moveit2_tutorials/moveit2_tutorials.repos

5. Préparez le package MoveIt

   sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro humble –y

6. *Build le package*

   cd ..
   colcon build --mixin release

7. Installez la commande Cyclone DDS

   sudo apt install ros-humble-rmw-cyclonedds-cpp
   export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp

1.3.1.2 *Vérification*

1. Sourcez votre package MoveIt

   cd ~/ros2_ws

   source install/setup.bash

2. Exécutez la commande suivante

   ros2 launch moveit2_tutorials demo.launch.py rviz_tutorial:=false

En cas de reussite, RViz se lancera avec un robot Panda Arm dans l'espace de configuration (Figure 3).

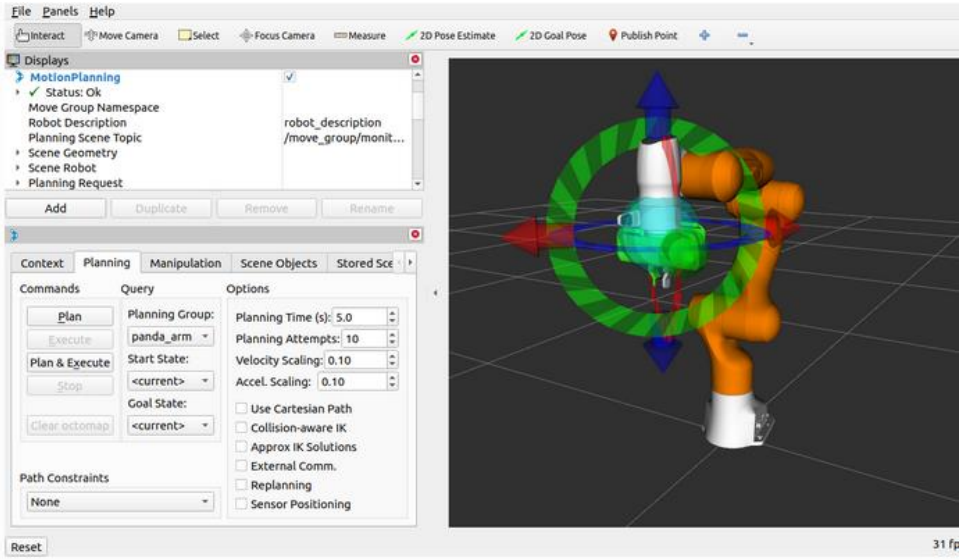

Figure 3: Panda Arm

1.4 Installation et utilisation du package Nav2

Le package Nav2 est un module de navigation ROS (https://navigation.ros.org). Ce package cherche à trouver un moyen de faire déplacer un robot mobile d'un point A à un point B. Il peut également être appliqué dans d'autres applications impliquant la navigation de robots, comme le suivi des objets dynamiques.

Suivez la procédure d'installation du package Nav2 telle que reprise dans la vidéo Nav2_installation_Tutorial.webm. Vérifiez le fonctionnement du Nav2 comme indiqué dans la vidéo suivante: ***Moveit_Tutorial.webm****. L'exercice va constituer à lancer un robot Turtlebot3 qui va effectuer la tâche de planification. Après avoir indiqué le point initial et le point cible, le robot va se rendre à la position de la cible en évitant des obstacles.*

Ressource: https://navigation.ros.org/getting_started/index.html

1.4.1 Tâches à exécuter

1.4.1.1 Installation

1. Installez le package Nav2

   sudo apt install ros-humble-navigation2
   sudo apt install ros-humble-nav2-bringup

2. Installez le robot turtlebot3

   *sudo apt install ros-humble-turtlebot3**

1.4.1.2 *Vérification*

1. Definissez les variables d'environnement suivantes

   export TURTLEBOT3_MODEL=waffle
   export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/humble/share/turtlebot3_gazebo/models

2. Lancez le robot turtlebot3 dans un environnement avec obstacles

   ros2 launch nav2_bringup tb3_simulation_launch.py headless:=False

En cas de réussite, RViz et Gazebo se lanceront avec un robot Turtlebot3 (Figure 4).

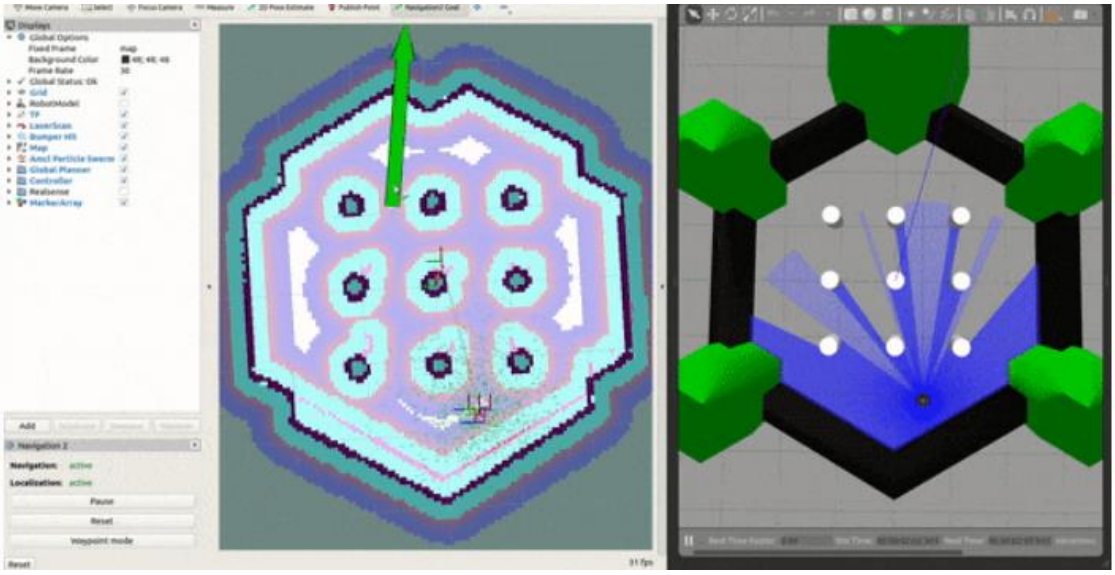

Figure 4: Turtlebot3 sur RViz (Gauche) et Gazebo (droite)

1.5 Installation et utilisation du package PlanSys2

ROS2 Planning System (PlanSys2 en abrégé) est un projet dont l'objectif est de fournir aux développeurs de robotique un système de planification fiable, simple et efficace basé sur PDDL (https://plansys2.github.io).

Suivez la procédure d'installation du package PlanSys telle que reprise dans la vidéo ***PlanSys2_installation_Tutorial.webm.*** *Vérifiez le fonctionnement du PlanSys2 comme indiqué vers la fin de la vidéo. L'exercice va constituer à lancer un robot Turtlebot3 qui va effectuer la tâche de planning control. Le robot va réaliser de différents plans afin d'atteindre des cibles.*

Ressource: https://plansys2.github.io/build_instructions/index.html

1.5.1.1 Installation

1. Installez le package PlanSys2

   *sudo apt install ros-humble-plansys2-**

2. Clonez les fichiers nécessaires du package

   mkdir -p ~/plansys2_ws/src
   cd ~/plansys2_ws/src
   git clone https://github.com/IntelligentRoboticsLabs/ros2_planning_system.git

   git clone https://github.com/IntelligentRoboticsLabs/plansys2_tfd_plan_solver.git
   git clone https://github.com/IntelligentRoboticsLabs/ros2_planning_system_examples.git

   cd ~/plansys2_ws
   rosdep install -y -r -q --from-paths src --ignore-src --rosdistro humble
   colcon build --symlink-install

1.5.1.2 *Vérification*

1. Sur un terminal, exécutez les commandes suivantes :

   cd ~/plansys2_ws
   source install/setup.bash
   export TURTLEBOT3_MODEL=waffle
   export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/humble/share/turtlebot3_gazebo/models
   ros2 launch plansys2_patrol_navigation_example patrol_example_launch.py headless:=False

2. Sur un autre terminal, exécutez la commande suivante :

   cd ~/plansys2_ws
   source install/setup.bash
   ros2 run plansys2_patrol_navigation_example patrolling_controller_node

En cas de réussite, RViz et Gazebo se lanceront avec un robot Turtlebot3 pour planning control (Figure 4).

2 Planification de trajectoires avec MoveIt

Ce laboratoire vous permet de vous familiariser avec l'outil MoveIt permettant au robots de se déplacer en évitant des obstacles. MoveIt invoque OMPL, la librairie mentionnée dans le cours. Pour ce laboratoire, vous allez vous concentrer sur MoveIt avec l'algorithme de OMPL choisi par défaut par MoveIt.

2.1 Panda Arm, PR2 et Turtlebot3

La première tâche pratique du devoir consiste à exécuter des requêtes de planification pour les robots Panda Arm (Figure 1) , PR2 (Figure 2) et Turtlebot3 (Figure 3) dans un environnement parsemé d'obstacles. Panda Arm a été développé par Franka Emika, une société allemande fondée en 2016 spécialisée dans la fabrication de robots collaboratifs. PR2 est un robot qui combine la mobilité pour naviguer dans les environnements humains et la dextérité pour saisir et manipuler des objets dans ces environnements. Turtlebot3 est un robot mobile nouvelle génération modulable, compacte et personnalisable.

Figure 1: Panda Arm

Figure 2: PR2

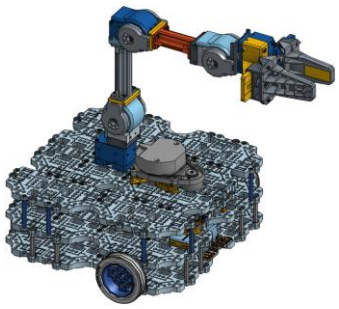

Figure 3: Turtlebot3

2.1.1 Format URDF

URDF est un format XML permettant de représenter des modèles de robots. Les chaînes cinématiques y sont modélisées à l'aide entre autres de liaisons (*joints*) et de pièces (*links*). Un exemple est représenté sur la Figure 3.

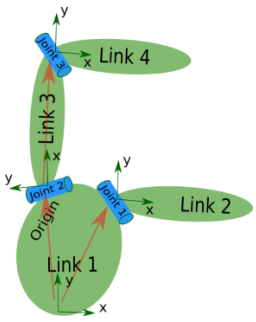

Figure 4: Exemple de chaîne cinématique

2.1.2 Tâches à réaliser

Contrairement aux exercices précédents, aucun package vous permettant d'utiliser directement le robot ne vous est fourni. Vous devez utiliser l'outil *MoveIt! Setup Assistant* pour créer ce package de configuration pour les trois robots.

1. Pour créer le package de configuration, il est nécessaire d'avoir un fichier de type *urdf* pour le robot Panda Arm. Le fichier se trouve à l'endroit suivant :

   ~/ros2_ws/src/moveit_resources/panda_description/urdf/panda.urdf

   Exécutez la commande suivante pour construire le package

   colcon build - -mixin release

   À l'aide du *MoveIt! Setup Assistant*, générez un package nommé *panda_moveit_generated* dans le répertoire *src/* de l'espace de travail r*os2_ws*.

   cd ros2_ws
   source install/setup.bash
   ros2 launch moveit_setup_assistant setup_assistant.launch.py

   Le tutoriel suivant peut servir de guide pour cette tâche :
   https://moveit.picknik.ai/humble/doc/examples/setup_assistant/setup_assistant_tutorial.html

   Bien que ce tutoriel s'applique au robot *Panda Arm* de la compagnie *Franka Emika*, la démarche est sensiblement la même pour les robots PR2 et Turtlebot3. Il y a quelques différences à prendre en considération :

   - Lors de la création des *Planning Groups* pour les bras, utilisez pour le bras droit et pour le bras gauche un *Chain Group* (bouton *Add Kin. Chain*).
   - L'ajout des Planning Groups pour les mains du robot est optionnel. De ce fait, l'ajout des *Ends Effectors* est aussi optionnel.
   - Etc.

2. Visualisez le robot dans RViz en utilisant le package *panda_moveit_generated* généré :
   ros2 launch turtlebot3_moveit_generated demo.launch

   Il est possible qu'aucun marqueur interactif ne soit affiché.

3. Ajoutez des objets sur la table qui se situe devant le robot (des bouteilles par exemple). Effectuez des requêtes de planification impliquant l'évitement de collision entre les bras du robot et ces objets. Le tutoriel suivant peut servir de guide pour cette tâche :
   https://moveit.picknik.ai/main/doc/tutorials/planning_around_objects/planning_around_objects.html

4. Reprendre les étapes précédentes pour les robots PR2 et turtlebot3. Les packages seront nommés *pr2_moveit_generated* et *turtlebot3_moveit_generated* . Le fichier urdf pour PR2 se trouve à l'endroit suivant :

~/ros2_ws/src/moveit_resources/pr2_description/urdf/robot.xml

Pour Turtlebot3, le fichier se trouve à l'endroit suivant du répertoire *https://github.com/ROBOTIS-GIT/turtlebot3_manipulation*

turtlebot3_manipulation/turtlebot3_manipulation_description/urdf/turtlebot3_manipulation.urdf.xacro

Il faut commencer par cloner ce répertoire dans *workspace* nommé *turtlebot3_ws/src* (considérez la branche *foxy-humble*)

Rassurez-vous qu'il n'y a pas d'erreur. D'autres répertoires pourraient être requis pour une construction sans erreur. Le tutoriel suivant peut servir de guide pour cette tâche : https://emanual.robotis.com/docs/en/platform/turtlebot3/manipulation/#manipulation

3 Déplacement des objets

Pour ce laboratoire, vous allez programmer un robot pour déplacer des objets d'un endroit vers un autre en utilisant l'outil MoveIt. Ceci sera implémenté dans les packages générés à l'étape précédente (*panda_moveit_generated*, *pr2_moveit_generated* et *turtlebot3_moveit_generated)*. Les étapes à compléter sont résumées dans la Figure 5.

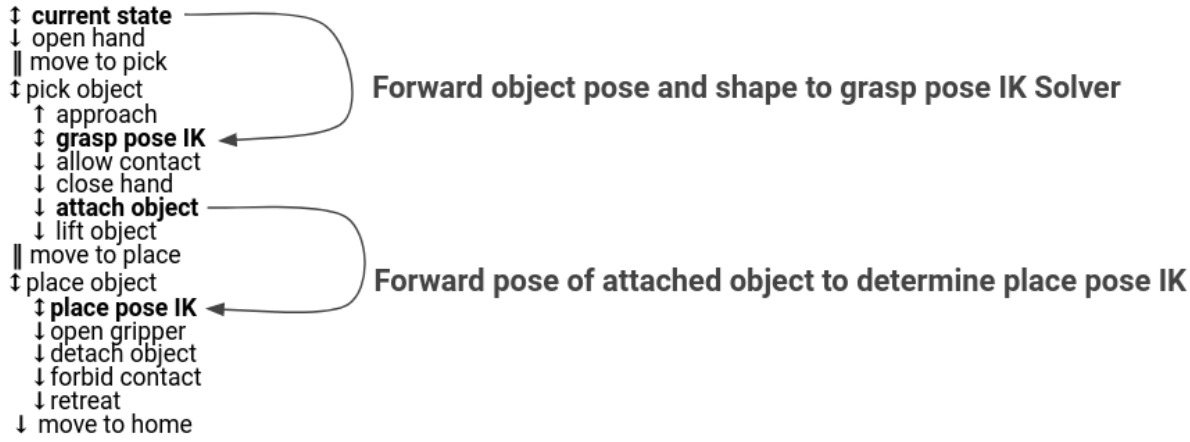

Figure 5

3.1 Tâches à réaliser

Votre objectif est de se servir des trois packages créés dans l'étape précédente et implémenter le déplacement des objets par les robots.

1. Clonez-le package suivant dans le *workspace* de MoveIt

   git clone https://github.com/ros-planning/moveit_task_constructor.git -b ros2

2. Créez un nouveau package en exécutant la commande suivante :

   ros2 pkg create --build-type ament_cmake --node-name move_object_robot move_object_robot

3. Ajoutez les stages comme indiqué dans la Figure 5.

4. Testez avec de différents algorithmes de planning. Ces algorithmes se trouvent sous l'onglet *Context* du Rviz.

Le tutoriel suivant peut servir de guide pour cette tâche :
https://moveit.picknik.ai/humble/doc/tutorials/pick_and_place_with_moveit_task_constructor/pick_and_place_with_moveit_task_constructor.html

4 Planification de tâches avec PlanSys2

Ce laboratoire va vous familiariser avec *PlanSys*, un planificateur de tâches utilisant le langage PDDL (pour représenter le modèle de l'environnement, autrement dit, représenter les actions du robot dans l'environnement) que vous avez vu en classe. PlanSys permet au robot de planifier et exécuter le plan.

4.1 Tâches à réaliser

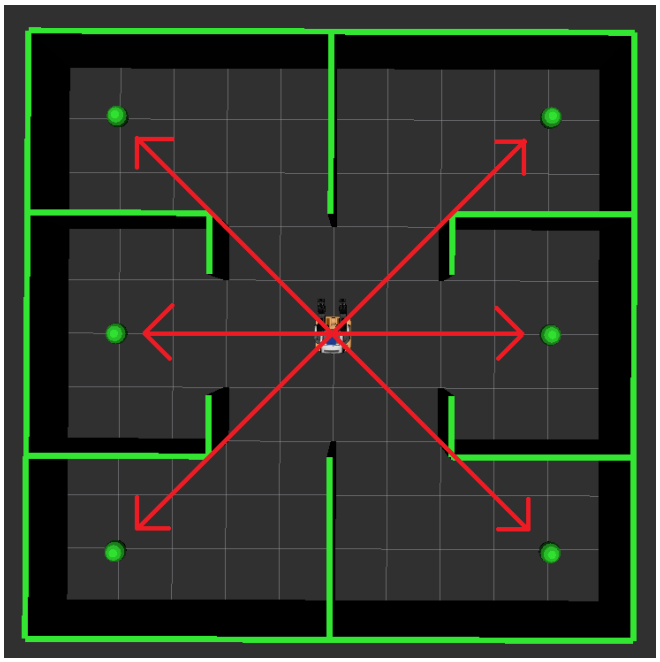

Figure 5: Planning

1. Ajoutez de 4 waypoints afin d'avoir une configuration similaire à celle dans la Figure 6 dans le fichier suivant :

   ros2_planning_system_examples/plansys2_patrol_navigation_example/src/move_action_node.cpp

2. Modifiez le PDDL dans le fichier suivant afin de prendre en compte la reconfiguration de l'environnement :

   ros2_planning_system_examples/plansys2_patrol_navigation_example/src/patrolling_controller_node.cpp

3. Exécutez la commande suivante sur un terminal :

   ros2 launch plansys2_patrol_navigation_example patrol_example_launch.py

4. Exécutez la commande suivante sur un autre terminal :

   ros2 run plansys2_patrol_navigation_example patrolling_controller_node

Vous devrez observer le robot exécuter un plan automatiquement. Le tutoriel suivant peut servir de guide pour cette tâche : https://plansys2.github.io/tutorials/docs/controller_example.html#tutorial-steps