Exercices ROS 2 – URDF XACRO

1. Introduction

Le but de l'exercice est de voir l'utilisation de XACRO et du language URDF pour remplacer/intégrer un autre robot sur la plateforme Caspr.

Il comprend :

- L'utilisation de RVIZ et du Robot_State_Publisher pour voir le robot sur une interface graphique et de la manipuler axe par axe.
- L'intégration d'un robot UR5e à la place du UR10e dans une scène préconstruite.

2. L'URDF de la scène de base

L'URDF de la scène du workshop intègre :

- Un bras UR10e
- Une bloc « Camera » en bout de bras
- Une plateforme « cube » qui a été placé pour sur-élevé le bras

Copiez le package qui contient l'URDF dans le workspace. Compiler et sourcer :

```
$ cp -r ~/workshop_rosconfr_2025/workshop_robot_description ~/my_ros_ws/src
$ cd ~/my_ros_ws
$ rosdep install —from-paths src –y –r —ignore-src //Cette commande permet d'installer les dépendances des sources
$ colcon build
$ source install/setup.bash
```

Allez ouvrir le fichier URDF de la plateforme :

```
$ cd ~/my_ros_ws/src/workshop_robot_description/urdf/
$ code .  // ouvre l'IDE code dans le dossier actuel (Changer code par votre IDE)
```

Ouvrez le fichier **ur10e_with_cube.urdf.xacro**

Le contenue de l'URDF est le suivant :

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="workshop">

   <link name="world"/>

    <link name="cube_link">
    <visual>
```

```
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
        <box size="0.5 0.5 0.5"/>
        </geometry>
        <material name="gray">
        <color rgba="0.6 0.6 0.6 1"/>
        </material>
    </visual>
    <collision>
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <geometry>
        <box size="0.5 0.5 0.5"/>
        </geometry>
    </collision>
    <inertial>
        <mass value="2.0"/>
        <inertia  ixx="0.02"  ixy="0.0"  ixz="0.0"  iyy="0.02"  iyz="0.0"
izz="0.02"/>
    </inertial>
    </link>

    <joint name="world_to_cube" type="fixed">
        <parent link="world"/>
        <child link="cube_link"/>
        <origin xyz="0 0 0.25" rpy="0 0 0"/>
    </joint>
    <!--Alternative cube link for an elevator-->
    <!--
    <joint name="world_to_cube" type="prismatic">
        <parent link="world"/>
        <child link="cube_link"/>
        <origin xyz="0 0 0.25" rpy="0 0 0"/>
        <axis xyz="0 0 1" rpy="0 0 0"/>
        <limit upper="3" lower="0"/>
    </joint>
    -->

    <xacro:include              filename="$(find              ur_descrip-
tion)/urdf/ur_macro.xacro"/>
    <xacro:include       filename="$(find       workshop_robot_descrip-
tion)/urdf/ur.ros2_control.xacro" />

    <xacro:arg name="ur_type" default="ur10e"/>

    <!-- parameters for ur -->
    <xacro:arg name="tf_prefix" default="" />
    <xacro:arg  name="joint_limit_params"  default="$(find  ur_descrip-
tion)/config/$(arg ur_type)/joint_limits.yaml"/>
```

```
    <xacro:arg   name="kinematics_params"   default="$(find   ur_descrip-
tion)/config/$(arg ur_type)/default_kinematics.yaml"/>
    <xacro:arg    name="physical_params"    default="$(find    ur_descrip-
tion)/config/$(arg ur_type)/physical_parameters.yaml"/>
    <xacro:arg    name="visual_params"    default="$(find    ur_descrip-
tion)/config/$(arg ur_type)/visual_parameters.yaml"/>
    <xacro:arg name="transmission_hw_interface" default=""/>
    <xacro:arg name="safety_limits" default="false"/>
    <xacro:arg name="safety_pos_margin" default="0.15"/>
    <xacro:arg name="safety_k_position" default="20"/>
    <xacro:arg name="force_abs_paths" default="false" />

    <xacro:ur_robot
        name="ur10e"
        tf_prefix="$(arg tf_prefix)"
        parent="cube_link"
        joint_limits_parameters_file="$(arg joint_limit_params)"
        kinematics_parameters_file="$(arg kinematics_params)"
        physical_parameters_file="$(arg physical_params)"
        visual_parameters_file="$(arg visual_params)"
        safety_limits="$(arg safety_limits)"
        safety_pos_margin="$(arg safety_pos_margin)"
        safety_k_position="$(arg safety_k_position)"
        force_abs_paths="$(arg force_abs_paths)"
        >
        <origin xyz="0 0 0.25" rpy="0 0 0" />
    </xacro:ur_robot>



    <link name="camera_link">
    <visual>
        <origin xyz="0.05 0 0.06" rpy="0 0 0"/>
        <geometry>
        <box size="0.06 0.03 0.04"/>
        </geometry>
        <material name="black">
        <color rgba="0.05 0.05 0.05 1"/>
        </material>
    </visual>
    </link>

    <joint name="ee_to_camera" type="fixed">
        <parent link="tool0"/>
        <child link="camera_link"/>
        <origin xyz="0.05 0 0.06" rpy="0 0 0"/>
    </joint>

    <xacro:ur_ros2_control
```

```
        name="ur10e"
        tf_prefix="$(arg tf_prefix)"
        kinematics_parameters_file="$(arg kinematics_params)"
        transmission_hw_interface="$(arg transmission_hw_interface)"
  />

    </robot>
```

Les différents éléments :

```
<link name="world"/>
```

Cette ligne rajoute un link nommé world. Ce link servira de référence aux autre links. C'est un peut notre origine.

On intègre le support pour le bras comme suit :.

```
<link name="cube_link">
<visual>
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <geometry>
  <box size="0.5 0.5 0.5"/>
  </geometry>
  <material name="gray">
  <color rgba="0.6 0.6 0.6 1"/>
  </material>
</visual>
<collision>
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <geometry>
  <box size="0.5 0.5 0.5"/>
  </geometry>
</collision>
<inertial>
  <mass value="2.0"/>
  <inertia ixx="0.02" ixy="0.0" ixz="0.0" iyy="0.02" iyz="0.0" izz="0.02"/>
</inertial>
</link>
```

On crée un link auquel on ajoute des éléments des visuels et de collisions. Ici c'est une box (un élement géométrique simple) de 50cm de côté. On y ajoute également une texture pour le visuel. Pour l'inertie, elle ne sers que pour des simulations (ce que l'on ne fera pas dans ce workshop).

On ajoute un joint (lien de référence) entre le world et le cube_link.

```
<joint name="world_to_cube" type="fixed">
  <parent link="world"/>
  <child link="cube_link"/>
  <origin xyz="0 0 0.25" rpy="0 0 0"/>
</joint>
```

Ce lien est fixe et est placer à 25cm en hauteur du centre de la scène.

On ajoute ensuite le robot :

```
  <xacro:include filename="$(find ur_description)/urdf/ur_macro.xacro"/>
  <xacro:include filename="$(find workshop_robot_description)/urdf/ur.ros2_control.xacro" />

  <xacro:arg name="ur_type" default="ur10e"/>

  <!-- parameters for ur -->
  <xacro:arg name="tf_prefix" default="" />
  <xacro:arg name="joint_limit_params" default="$(find ur_description)/config/$(arg ur_type)/joint_limits.yaml"/>
  <xacro:arg name="kinematics_params" default="$(find ur_description)/config/$(arg ur_type)/default_kinematics.yaml"/>
  <xacro:arg name="physical_params" default="$(find ur_description)/config/$(arg ur_type)/physical_parameters.yaml"/>
  <xacro:arg name="visual_params" default="$(find ur_description)/config/$(arg ur_type)/visual_parameters.yaml"/>
  <xacro:arg name="transmission_hw_interface" default=""/>
  <xacro:arg name="safety_limits" default="false"/>
  <xacro:arg name="safety_pos_margin" default="0.15"/>
  <xacro:arg name="safety_k_position" default="20"/>
  <xacro:arg name="force_abs_paths" default="false" />

  <xacro:ur_robot
    name="ur10e"
    tf_prefix="$(arg tf_prefix)"
    parent="cube_link"
    joint_limits_parameters_file="$(arg joint_limit_params)"
    kinematics_parameters_file="$(arg kinematics_params)"
    physical_parameters_file="$(arg physical_params)"
    visual_parameters_file="$(arg visual_params)"
    safety_limits="$(arg safety_limits)"
    safety_pos_margin="$(arg safety_pos_margin)"
    safety_k_position="$(arg safety_k_position)"
    force_abs_paths="$(arg force_abs_paths)"
    >
    <origin xyz="0 0 0.25" rpy="0 0 0" />
  </xacro:ur_robot>
```

Cette partie sera un peu plus développer dans la correction mais noter les éléments suivants :

```
  <xacro:include filename="$(find ur_description)/urdf/ur_macro.xacro"/>
  <xacro:include filename="$(find workshop_robot_description)/urdf/ur.ros2_control.xacro" />
```

On intègre 2 fichiers externes permettant de charger des macros XACRO et notament ur_macro qui contient les descriptions de « tout les UR »

```
  <xacro:arg name="ur_type" default="ur10e"/>
```

Cet argument permet de charger le fichier ur10e.

```
    parent="cube_link"
```

Permet de définir que la base de l'UR sera relié au cube_link

```
    <origin xyz="0 0 0.25" rpy="0 0 0" />
```

Permet de positionner l'UR à 25cm en hauteur sur le cube (donc à 50cm de l'origine).

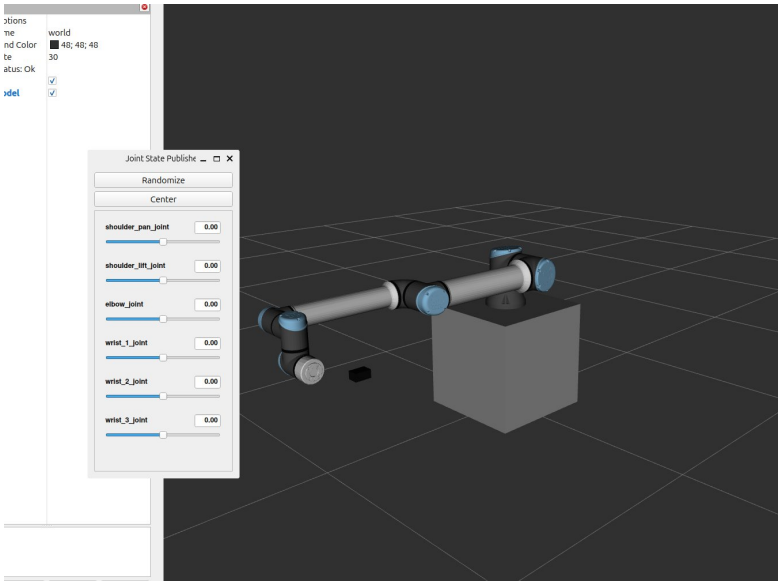
```
<link name="camera_link">
<visual>
   <origin xyz="0.05 0 0.06" rpy="0 0 0"/>
   <geometry>
   <box size="0.06 0.03 0.04"/>
   </geometry>
   <material name="black">
   <color rgba="0.05 0.05 0.05 1"/>
   </material>
</visual>
</link>

<joint name="ee_to_camera" type="fixed">
   <parent link="tool0"/>
   <child link="camera_link"/>
   <origin xyz="0.05 0 0.06" rpy="0 0 0"/>
</joint>
```

On ajoute un bloc camera (comme le cube) à la suite du l'UR (le dernier link de l'UR étant tool0)

Pour visualiser, l'URDF voici la commande :

```
$ ros2 launch workshop_robot_description display.launch.py
```



Essayer de déplacer les axes à l'aide de l'interfaces graphiques.

Nous allons maintenant chercher à connaître la position du TCP0 dans le monde. Dans un autre terminal:

```
$ ros2 run tf2_ros tf2_echo world tool0 // On cherche la position de tool0 (TCP0) dans le repère world.
```

Déplacer les différents axes pour voir la position évoluer.

Pour continuer :

- Essayer un autre type d'UR (UR5e, UR30 etc…)
- Essayer d'ajouter un autre bloc après tool0 (tout en gardant la caméra). Y ajouter des collisions. (Si vous voulez d'autres formes géométriques : https://wiki.ros.org/urdf/XML/link
- Modifier la liaison fixe entre le cube et la base robot par une autre (exemple dans l'URDF déjà présent) mais autres possibilités : https://wiki.ros.org/urdf/XML/joint