Universitat d'Alacant
Universidad de Alicante

Grado en Ingeniería Robótica:

Robots móviles

ROS: MoveIt

Manuel Herraiz Sala
Santiago Avendaño Flores
Manuel Pérez Gómez
Alexander Von Knoop Ruiz

Índice

Introducción

MoveIt! consiste en un conjunto de paquetes de software integrado en Robot Operating System (ROS) diseñado específicamente para modelar robots con brazos robóticos y planear trayectorias gracias a la cómoda configuración de los elementos del brazo robótico por medio de python o a la visualización y modificación de trayectorias por medio de una interfaz interactiva.

A su vez, dado que ROS puede tomar información de un gran número "topics" como sensores de distancia tipo láser o cámaras 3D, MoveIt! dispone de herramientas suficientes para construir una representación de su entorno para generar trayectorias de movimiento más eficaces y seguros para los operarios que se encuentren en su espacio de trabajo mientras sus movimientos son supervisados por el propio software para adaptarse a posibles cambios repentinos del entorno. [1]

Originalmente estaba diseñado para su aplicación en robots del grupo Personal Robot 2 (PR2), pero actualmente está disponible para aquellos robots que dispongan en ROS de su Unified Robot Description Format (URDF), del cual se detallará más adelante. En nuestro caso, nos centraremos en la descripción de su uso para el IRB 120 de ABB.

URDF

Como se ha explicado anteriormente, se va a explicar como configurar y usar la herramienta moveit y para ello se hará uso de un robot creado con el formato URDF (*Unified Robot Description Format*). En este formato, se programa en XML, y se definen todos los eslabones (o *links*) y todas las articulaciones (o *joints*) de nuestro robot, en este caso se va a simular un IRB 120 de ABB, con 6 grados de libertad más dos para controlar un efector final.

- Link: Para cada eslabón se debe especificar su nombre, su *inertial* (opcional), es decir, las propiedades inerciales de la articulación. Su *visual* (opcional), haciendo referencia a cómo debe de ser visualmente el objeto, su tamaño, forma etc. Y su *colision* (opcional), es decir, el área de colisión que tiene cada eslabón.

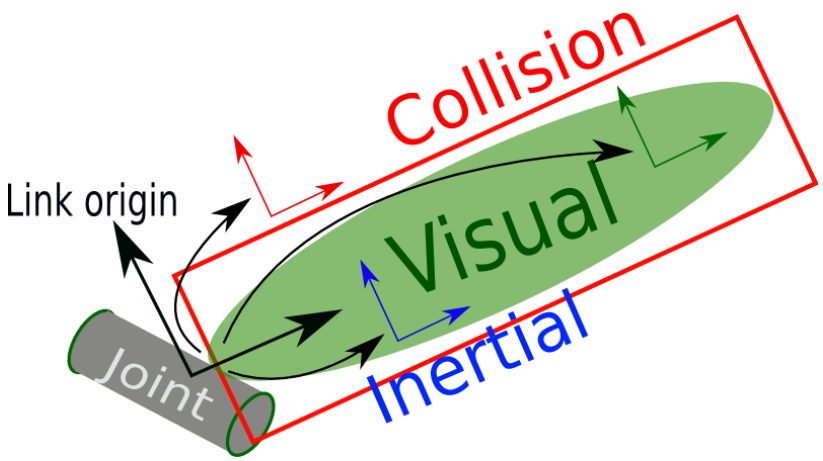

Imagen 1:

· *Inertial:* En este apartado se debe de especificar el origen de coordenadas del sistema inercial, el cual debe estar situado en el centro de masas de la figura y tiene que estar representado por las coordenadas cartesianas xyz y la orientación representada por roll, pitch y yaw. Además tiene que contener la masa del eslabón y la matriz de inercia de 3x3, la cual se obtendrá pasando los parámetros ixx, ixy, ixz, iyy, iyz, izz.

```
<inertial>
  <origin xyz="0 0 0.5" rpy="0 0 0"/>
  <mass value="1"/>
  <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
</inertial>
```

Imagen 2: Ejemplo de la inercia de un eslabón

· *Visual:* Igual que en el apartado anterior también se debe de especificar el sistema de coordenadas representado por xyz y roll pitch yaw, además se debe de especificar la geometría del eslabón y su tamaño, se puede elegir entre *box* en donde habrá que definir el tamaño de cada lado, *cylinder* en el cual se especificará el radio y la altura, *sphere* donde solo será necesario concretar el radio *y mesh* este es el más especial, ya que sirve para representar

imágenes en 3D que tengas del eslabón por lo tanto solo necesitará un *filename* y también se puede especificar una escala. Por último, se puede especificar el material, en el cual se puede especificar el color en escala rgba (red green blue alpha) y también se puede poner una textura especificando un *filename.*

```
<visual>
  <origin xyz="0 0 0" rpy="0 0 0" />
  <geometry>
    <box size="1 1 1" />
  </geometry>
  <material name="Cyan">
    <color rgba="0 1.0 1.0 1.0"/>
  </material>
</visual>
```

Imagen 3: Ejemplo de la parte visual de un eslabón

· *Colision:* En esta parte al igual que en las dos anteriores hay que especificar el sistema de coordenadas en xyz y roll pitch yaw, y también hay que especificar la geometria de colisión, que al igual que en el apartado anterior puede ser una caja, un cilindro, una esfera o un objeto más complejo.

```
<collision>
  <origin xyz="0 0 0" rpy="0 0 0"/>
  <geometry>
    <cylinder radius="1" length="0.5"/>
  </geometry>
</collision>
```

Imagen 4: Ejemplo de la parte colision de un eslabón

Para más información se puede ver en el enlace [2].

- Joint: Para cada articulación se debe especificar su nombre, su tipo de movimiento prismático, rotacional, fijo etc. El sistema de coordenadas representado por xyz y roll pitch yaw, el eslabón padre, es decir, cual es el link que transmite movimiento a la articulación, el eslabón hijo, el cual es el link al que el eslabón transmite el movimiento. El eje en el que se produce el movimiento (si el giro es en el eje z se especificará como <0 0 1>). El límite de velocidad, de articulación (posición máxima y mínima que puede alcanzar) y el límite de esfuerzo. También se pueden especificar más cosas, que se encuentran en [3], pero principalmente estás son las características básicas.

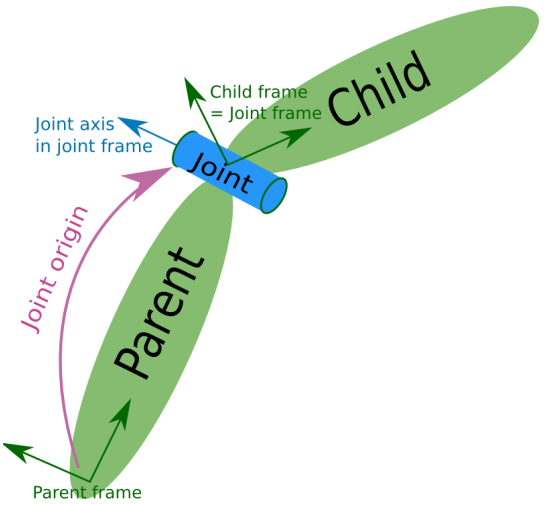

Imagen 5:

```
<joint name="link_00__link_01" type="revolute">
    <axis xyz="0 0 1" />
    <limit effort="1000" lower="${-(11*pi)/12}" upper="${(11*pi)/12}" velocity="0.5" />
    <origin rpy="0 0 0" xyz="0 0 ${l0}" />
    <parent link="link_00" />
    <child link="link_01" />
</joint>
```

Imagen 6: Ejemplo de una joint

Para simplificar y tener una estructura más flexible en la creación del archivo URDF se pueden usar macros, por ejemplo la que se puede ver en la siguiente imagen.

```
<xacro:macro name="m_link_cylinder" params="name origin_xyz origin_rpy radius length mass ixx ixy ixz iyy iyz izz">
    <link name="${name}">
        <visual>
            <origin rpy="${origin_rpy}" xyz="${origin_xyz}" />
            <geometry>
                <cylinder radius="${radius}" length="${length}" />
            </geometry>
        </visual>
        <collision>
            <origin rpy="${origin_rpy}" xyz="${origin_xyz}" />
            <geometry>
                <cylinder radius="${radius}" length="${length}" />
            </geometry>
        </collision>
        <inertial>
            <mass value="${mass}" />
            <origin rpy="${origin_rpy}" xyz="${origin_xyz}" />
            <inertia ixx="${ixx}" ixy="${ixy}" ixz="${ixz}" iyy="${iyy}" iyz="${iyz}" izz="${izz}" />
        </inertial>
    </link>
</xacro:macro>
```

Imagen 7: Macro de un eslabón cilíndrico

Una vez declarado todas las articulaciones y eslabones se puede mostrar en rviz, y con el nodo *joint_state_publisher* se pueden mover el robot sin haber creado el moveit.

```
<launch>

    <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find IRB_120)/urdf/robot_IRB120.urdf.xacro'

    <!-- Combine joint values -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
        respawn="false" output="screen">
    </node>

    <!-- Show in Rviz   -->
    <node name="rviz" pkg="rviz" type="rviz" args="-d $(find IRB_120)/launch/config.rviz"/>

    <!-- send joint values -->
    <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
        <param name="use_gui" value="true"/>
    </node>

</launch>
```

Imagen 8: Launch para lanzar el rviz y el joint_state_publisher

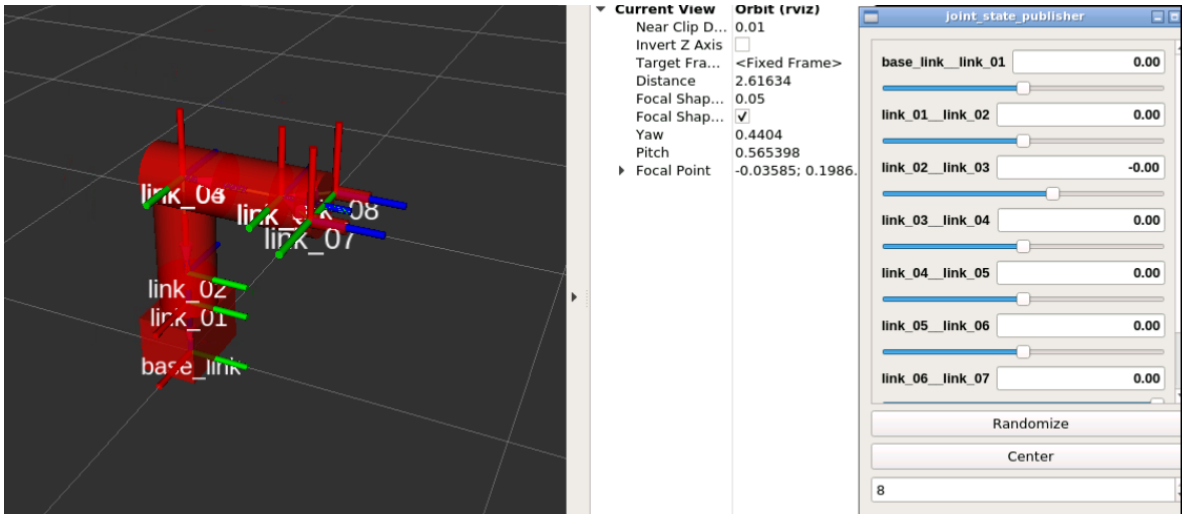

Imagen 9: Representación del robot en rviz y el nodo joint_state_publisher

MoveIt: planeador de trayectorias

Para poder usar la herramienta MoveIt con el robot creado anteriormente, en un terminal se ejecutará el siguiente archivo launch:

- roslaunch moveit_setup_assistant setup_assistant.launch

Este launch iniciará el asistente de MoveIt, en el cual se deberá introducir todos los datos del robot para así poder crear el paquete de configuración de MoveIt. Al lanzar el archivo launch aparecerá la siguiente ventana:

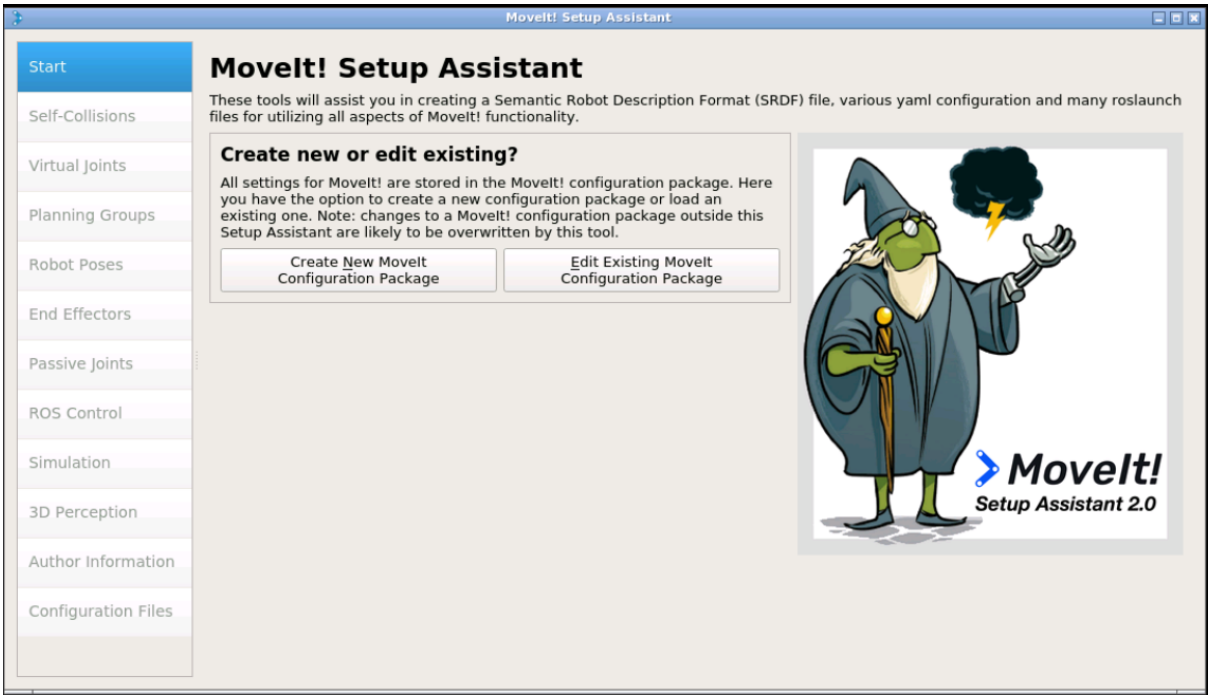

Imagen 10: Pantalla de inicio del asistente de paquetes de MoveIt .

En primer lugar, se debe generar la matriz de colisiones del robot, para ello se accede al apartado del asistente llamado "Self-Collisions"(Imagen 11), se escoge un valor de "Sampling Density" y finalmente se selecciona la opción llamada "Generate Collision Matrix". Tras esto ya se tendría la matriz de colisiones creada.

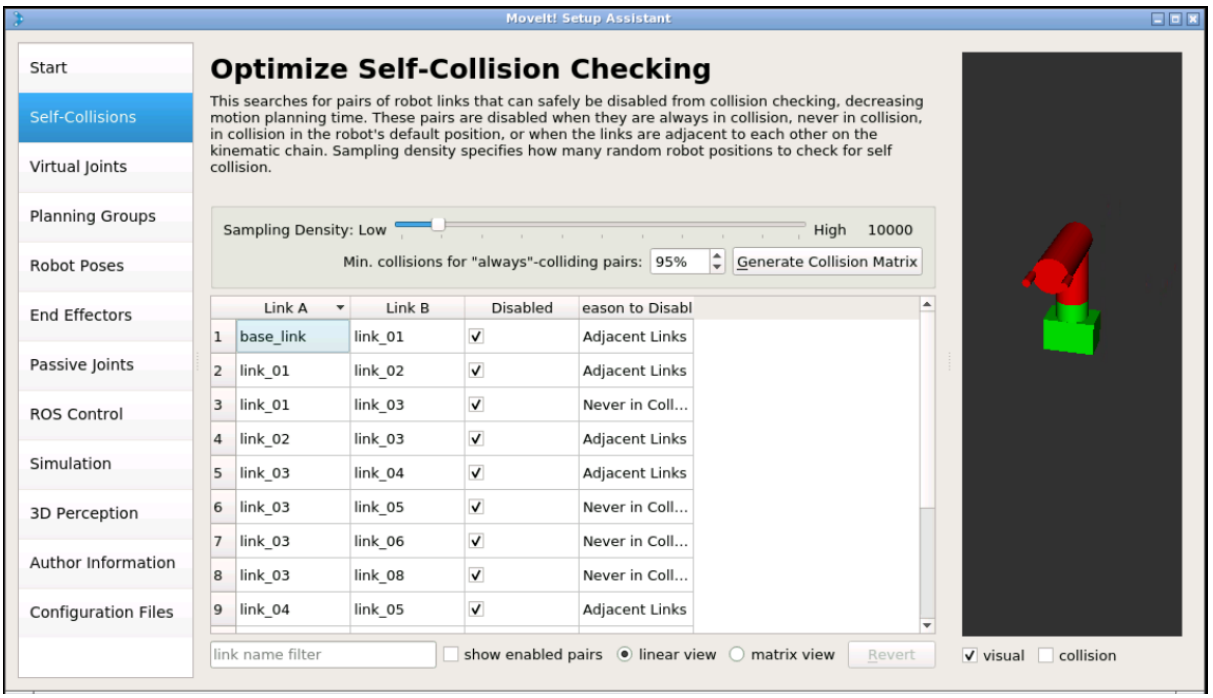

Imagen 11: Apartado "Self-Collisions" del asistente de MoveIt.

Después, se procede a crear una articulación virtual, para ello habrá que ir al apartado "Virtual joints" del asistente y en él nos aparecerá lo siguiente:

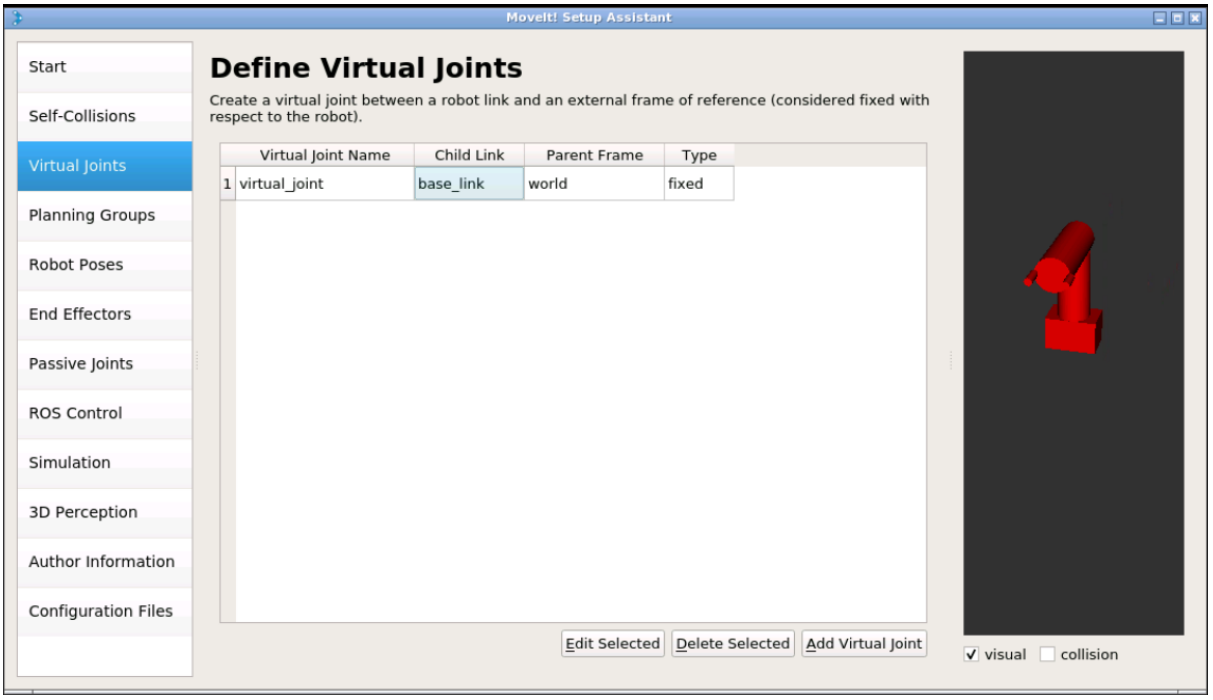

Imagen 12: Apartado "Virtual joints" del asistente de MoveIt.

Una vez que estamos en el apartado de articulaciones virtuales, se procederá a crear una. Para ello, se seleccionará la opción llamada "Add Virtual Joint" y nos aparecerá otra ventana en la cual se deberá introducir la siguiente información:

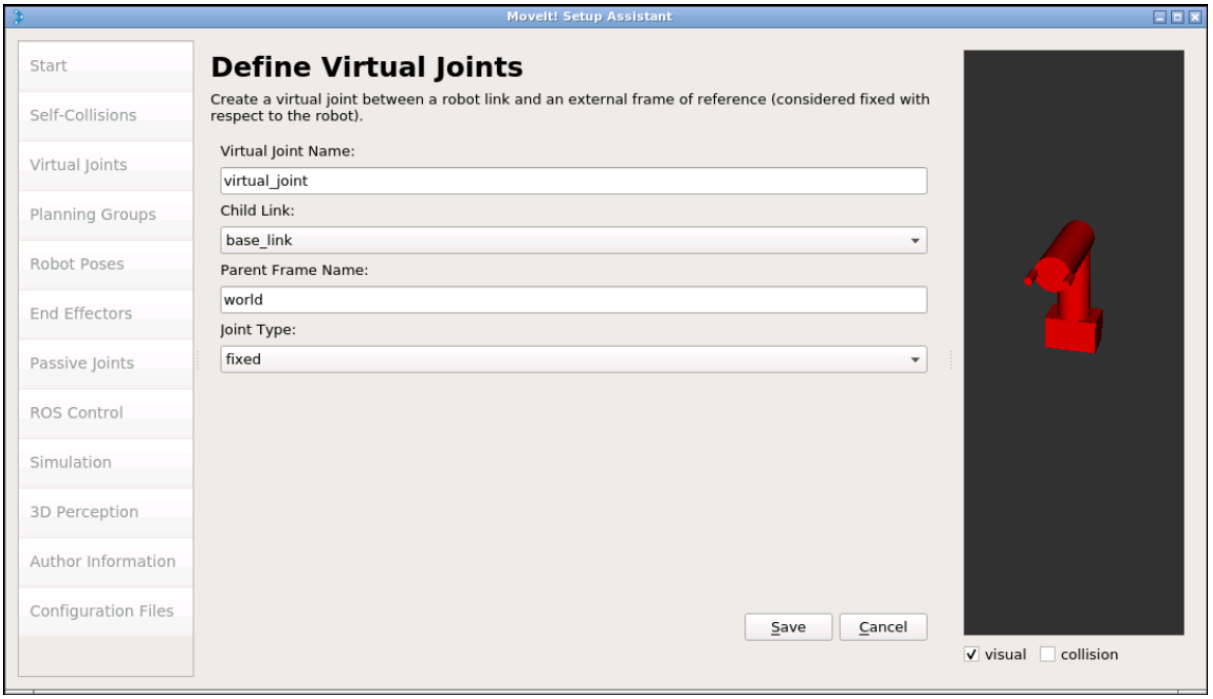

Imagen 13: Definición de una articulación virtual.

Para el caso de ejemplo utilizado, solo será necesario crear una, cuyos datos se pueden observar en la imagen 13. Posteriormente, será necesario definir los grupos articulares del robot, en el caso de los robots manipuladores, es común definir un grupo articular que

comprenda las articulaciones y eslabones del brazo robótico y otro grupo que defina los elementos del efector final. En la siguiente imagen se puede observar las opciones disponibles en la sección llamada "Planning Groups", sección en la cual se definen los grupos articulares:

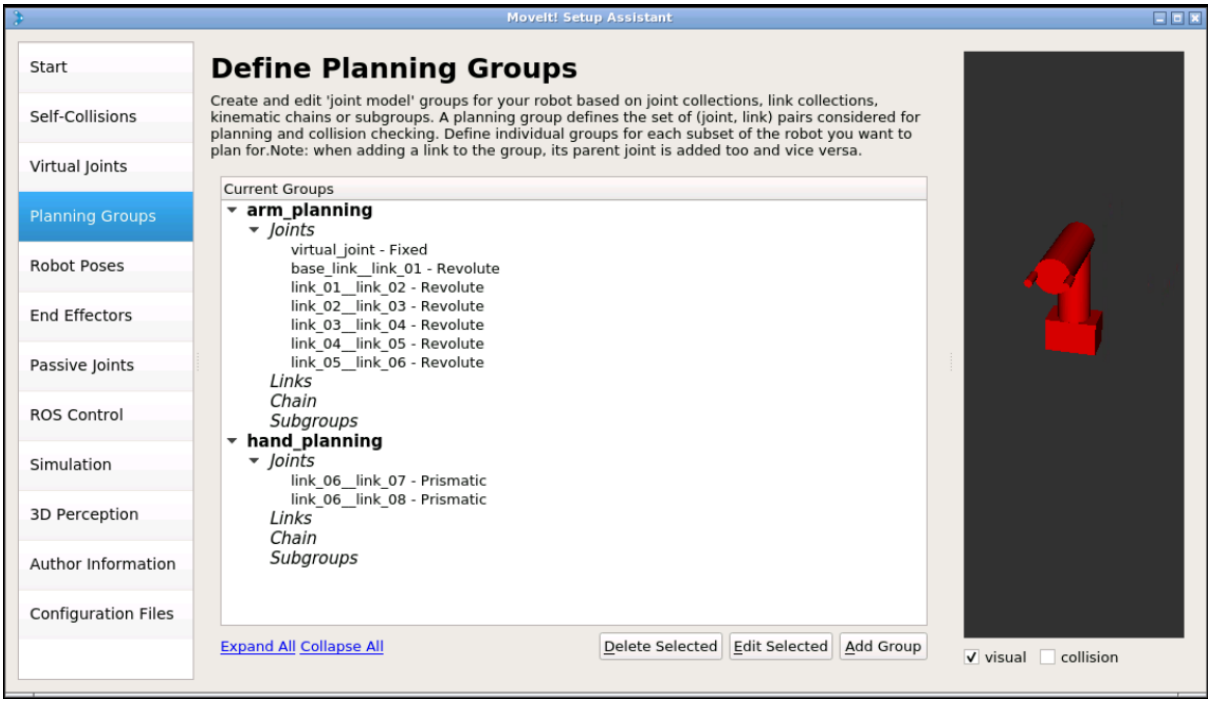

Imagen 14: Apartado "Planning Groups" del asistente de MoveIt.

Para añadir un grupo articular, se selecciona la opción "Add Group" y en ella se deberá rellenar una serie de datos como por ejemplo las articulaciones y eslabones del grupo, la elección del planificador,etc. Todo esto se puede observar en la imagen 15:

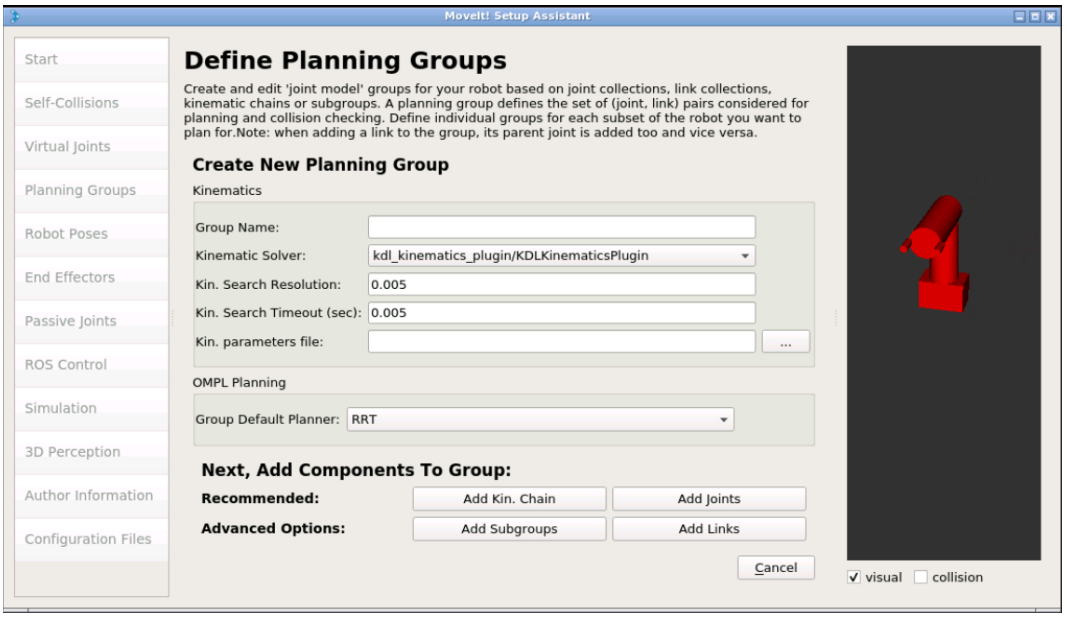

Imagen 15: Ventana de creación de un grupo articular.

Una vez ya definidos los grupos articulares, se establecerán una serie de posiciones conocidas del robot, dichas posiciones pueden ser "home", "all_zeros" o cualquier configuración articular que el robot pueda alcanzar. Para hacer esto, se entra en el apartado llamado "Robot Poses", mostrado en la imagen 16, y en él mediante la opción "Add Pose" se irán añadiendo posiciones articulares concretas. Al añadir una pose, en primer lugar se deberá seleccionar el grupo articular del cual se va definir la pose, después, se seleccionan los valores articulares de las articulaciones que componen el grupo y finalmente se le da un nombre a la pose y se guarda. Esto se puede observar en la imagen 17:

Imagen 16: Apartado "Robot Poses" del asistente de MoveIt.

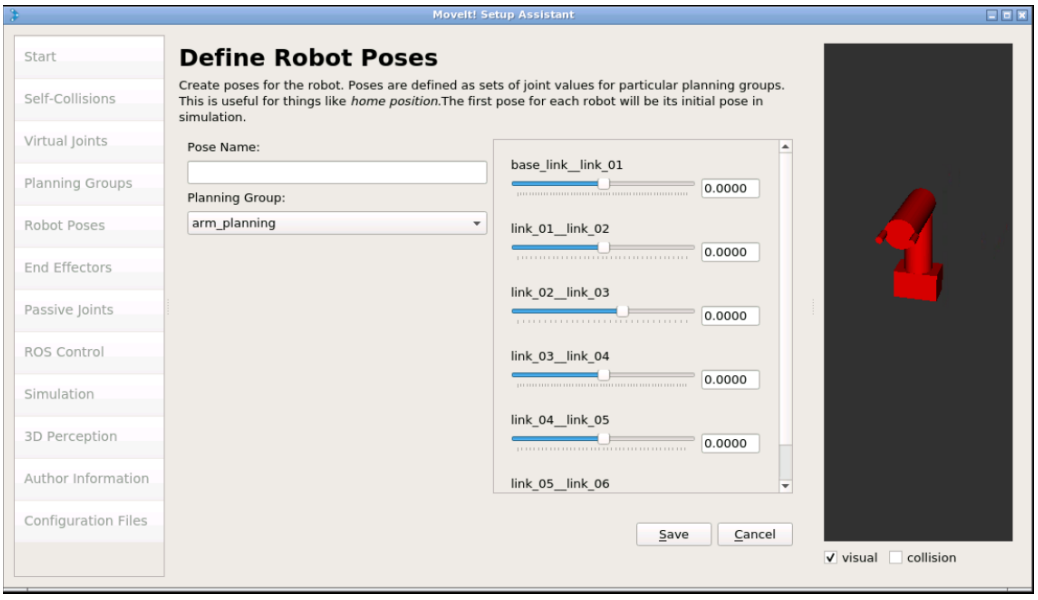

Imagen 17: Creación de una pose del robot concreta.

A continuación, se define el efector final del robot, esto se hará en la sección "End Effector"(mostrada en la imagen 18) del asistente, para definir un efector final será necesario indicar el grupo articular al que pertenece y el link padre(eslabón donde estará situado), esto se puede observar en la imagen 19:

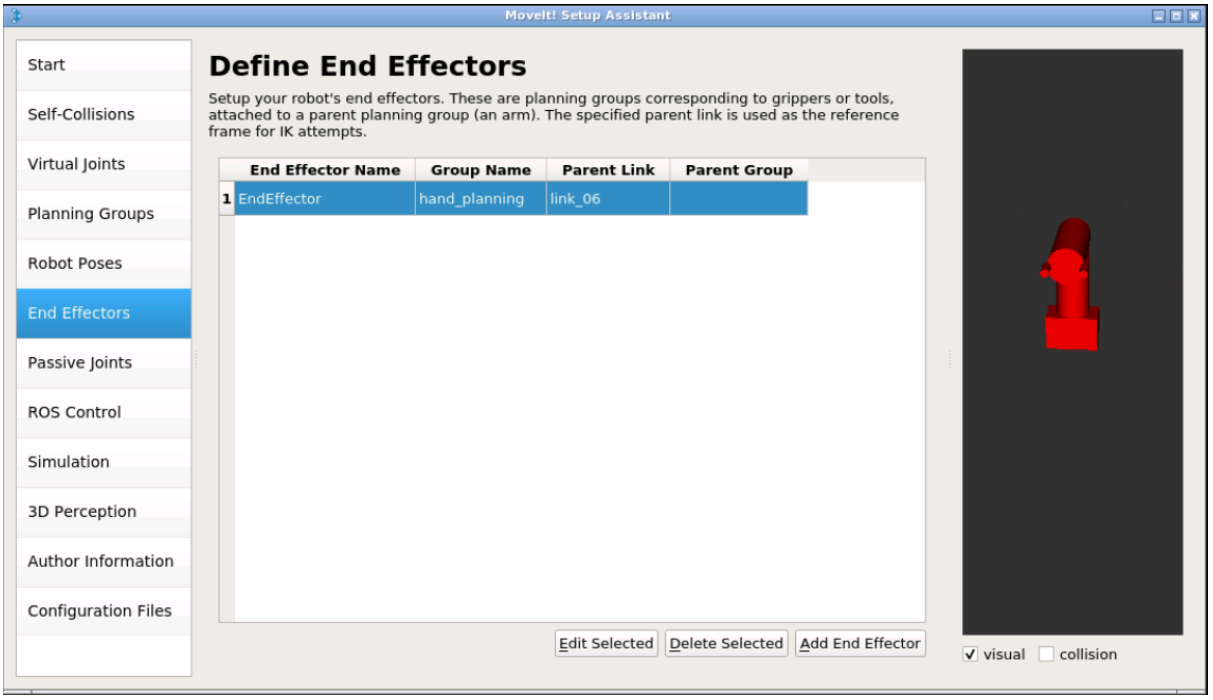

Imagen 18: Apartado "End Effectors" del asistente de MoveIt.

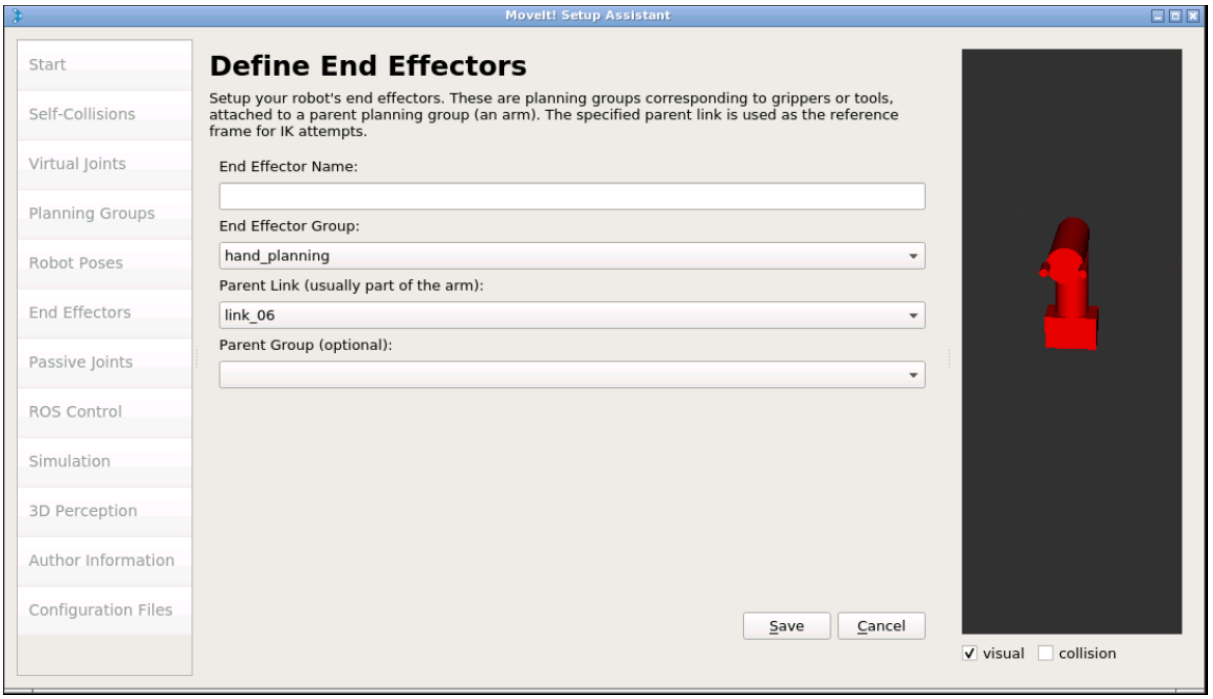

Imagen 19: Definición y creación de un efector final.

Luego de haber definido el efector final, se indica en la sección "Passive Joints"(Imagen 20) qué articulaciones son activas(articulaciones móviles) y cuales son pasivas(articulaciones fijas):

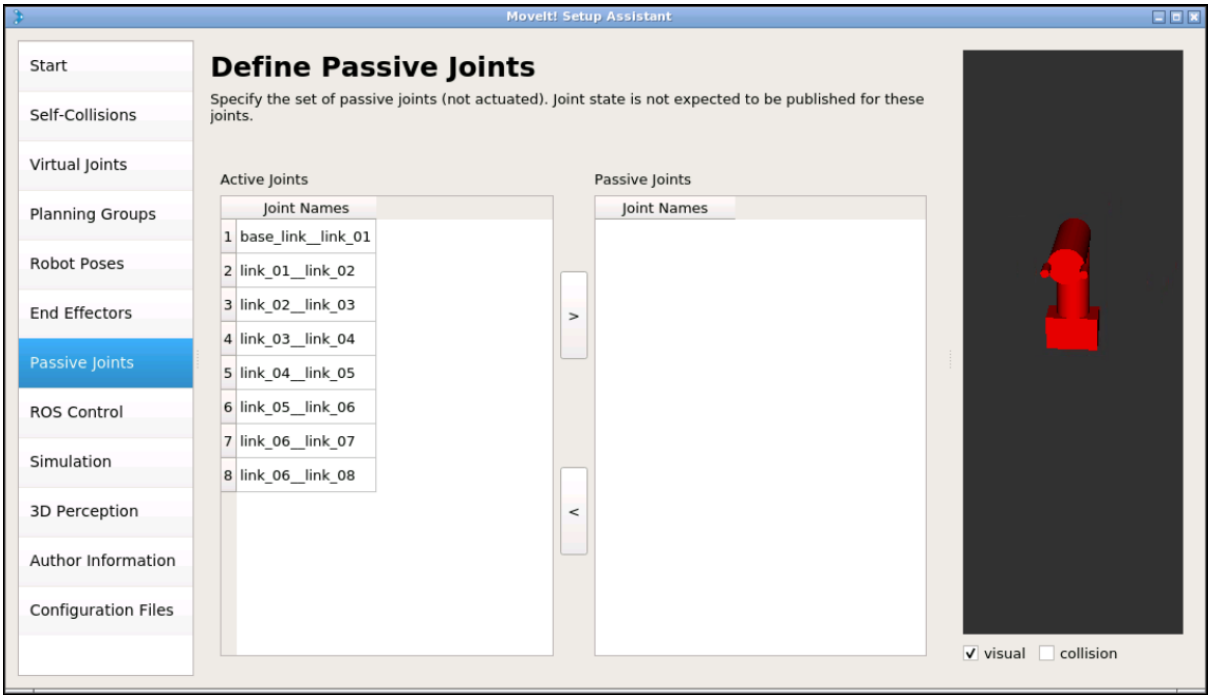

Imagen 20: Definición de las articulaciones activas y pasivas.

Más adelante se definen los controladores que tendrá el robot, para ello, en la sección "ROS Control" se añadirán los controladores necesarios para cada grupo articular, esto se puede hacer de manera manual o automática seleccionando la opción "Auto Add FollowJointsTrajectory Controllers For Each Planning Group"

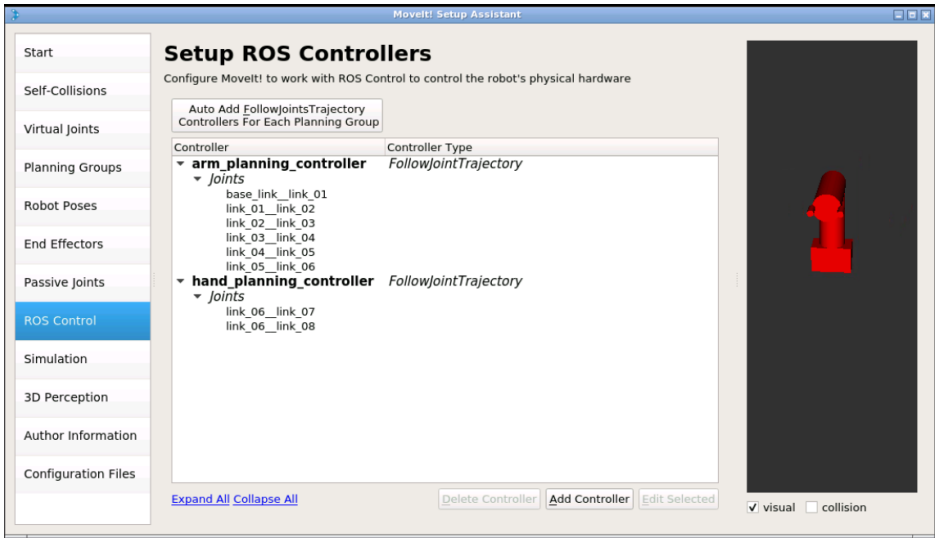

Imagen 21: Definición de las articulaciones activas y pasivas.

Después en la sección "3D Perception" se podrá seleccionar el plugin de un sensor 3D y configurar para su posterior uso, en el caso de ejemplo realizado no se ha utilizado ninguno.

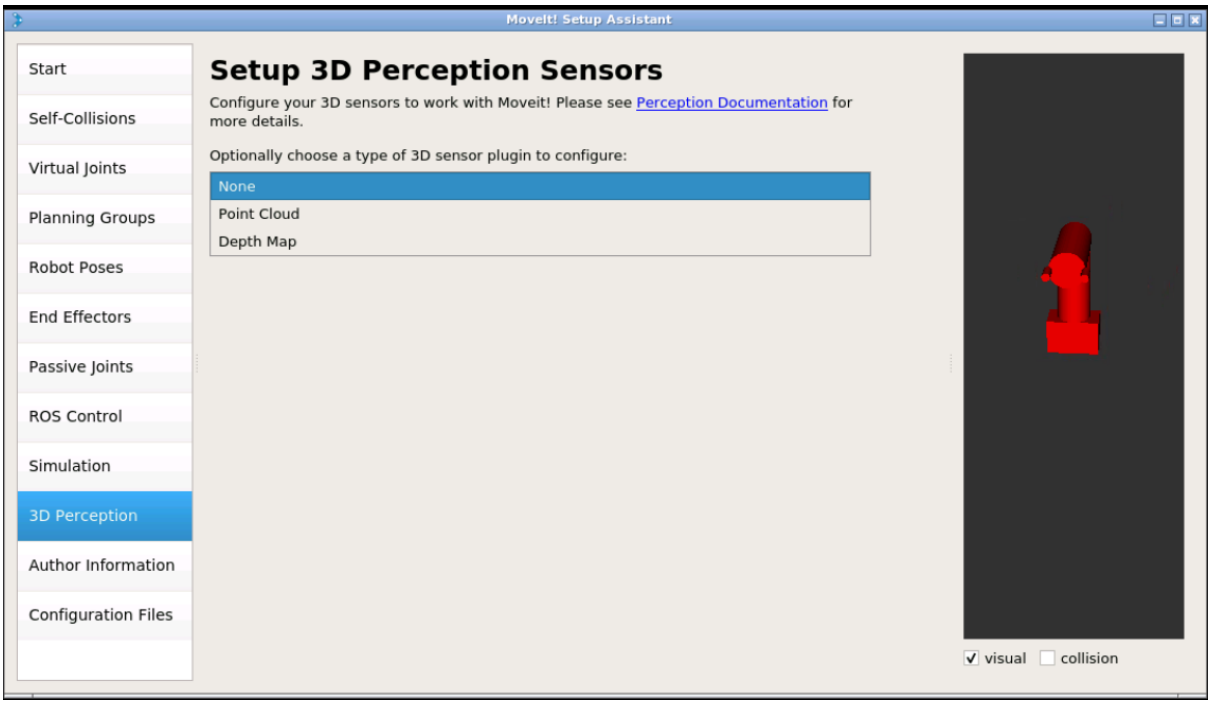

Imagen 22: Selección del sensor 3D que utilizará el robot.

Finalmente solo faltará rellenar el apartado de información del autor (Imagen 23) y el de archivos de configuración(Imagen 24), en el cual se seleccionará la ruta donde se quiere guardar el paquete y se seleccionará la opción "Generate Package":

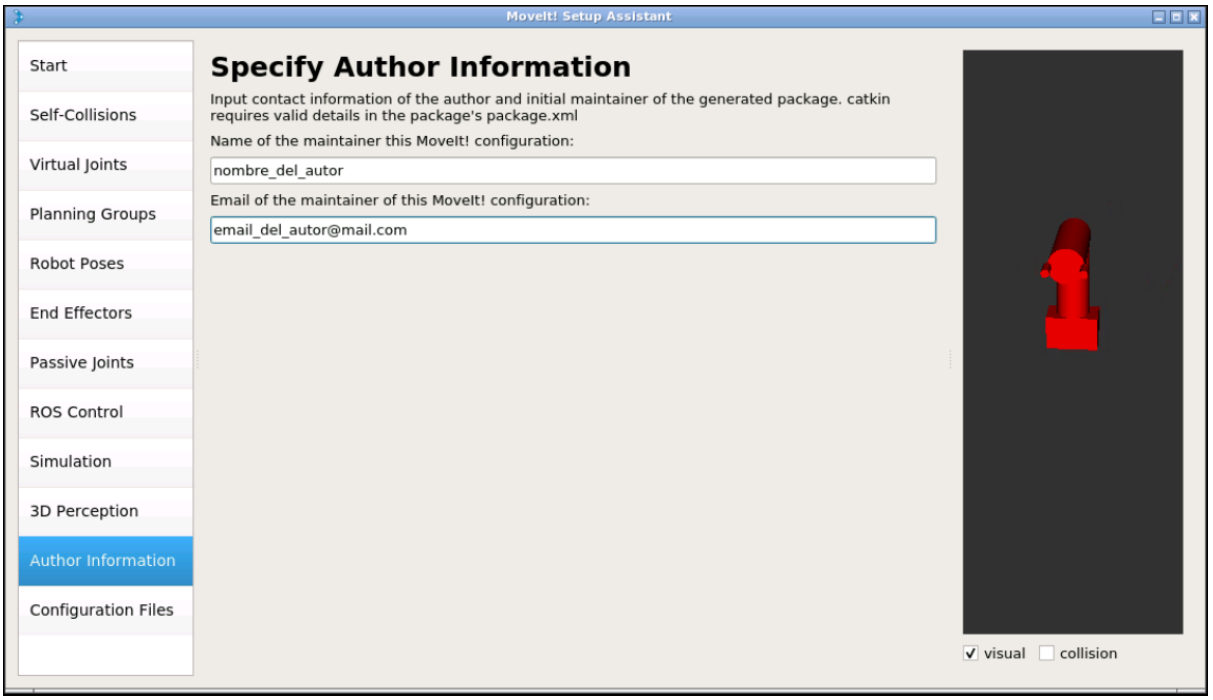

Imagen 23: Apartado de información del autor del asistente de MoveIt.

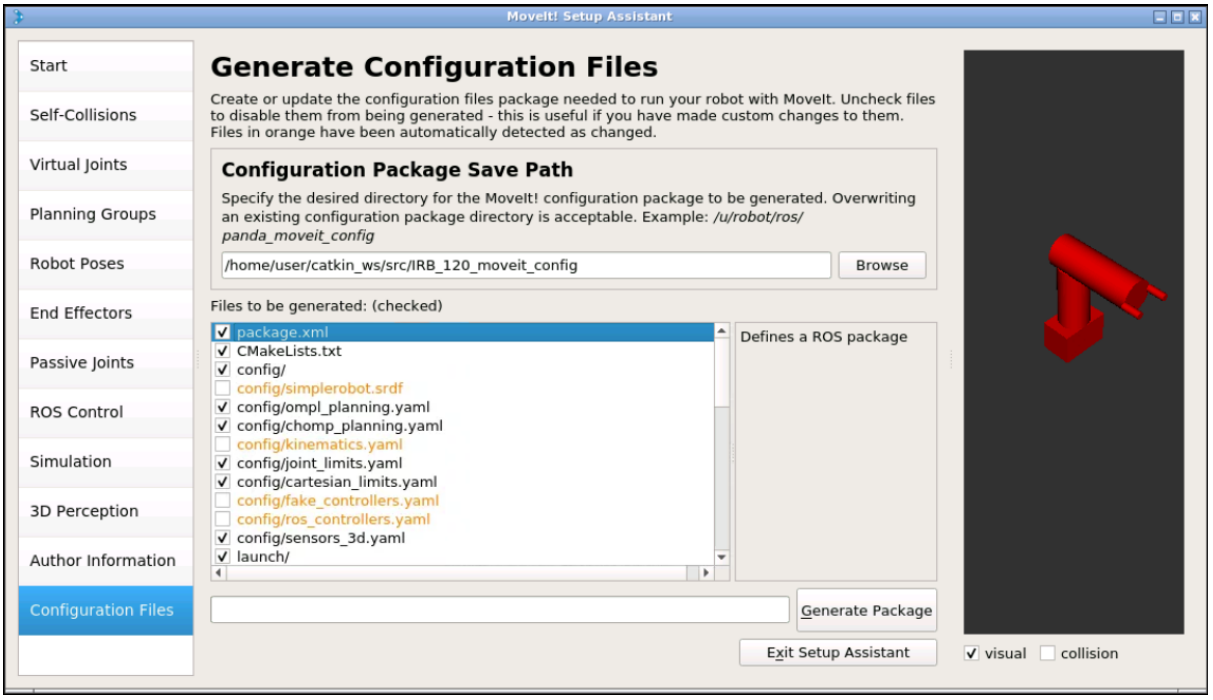

Imagen 24: Apartado "Configuration Files" del asistente de MoveIt.

Una vez creados todos los paquetes del MoveIt con la ayuda del *setup assistant*, se podría lanzar el *demo.launch* generado automáticamente y comenzar a probar cosas del MoveIt.

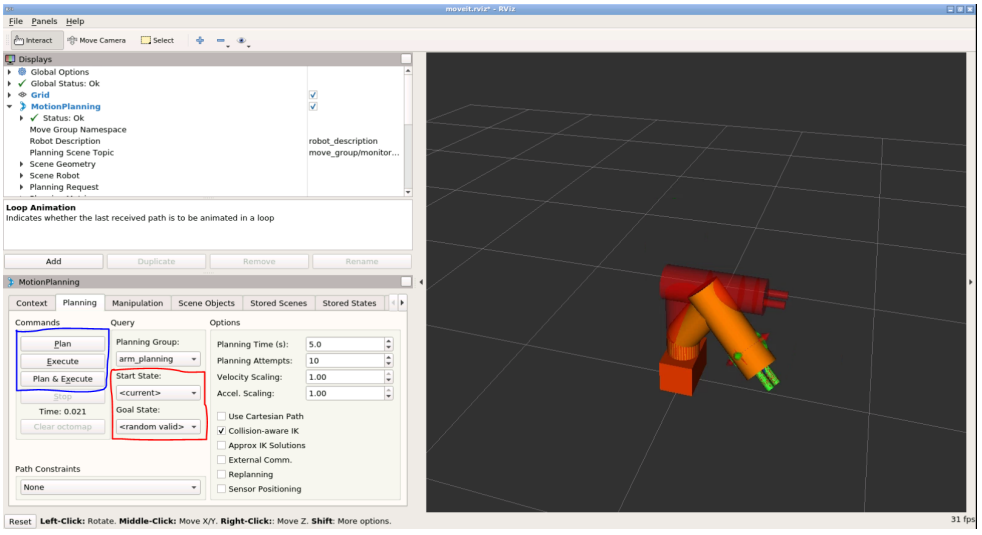

Imagen 25: Fichero demo.launch

Como se puede observar a la izquierda de la imagen se tiene el "panel" de control del MoveIt, en la parte marcada en rojo, se puede seleccionar de qué posición se desea partir (la posición actual) y hasta qué posición se quiere alcanzar (una posición aleatoria dentro del espacio de trabajo).

En la parte de la derecha, podemos observar al robot rojo transparente, el cual representa la posición actual del robot, y un robot naranja que es la posición objetivo, para hacer que se planifique la trayectoria y se ejecute se debe seleccionar a los botones señalados en azul, o bien se presiona el botón "plan" y luego el botón "execute" o se presiona directamente el botón "plan & execute", (una vez presionado el botón plan se mostrará una simulación del movimiento).

Además se puede seleccionar alcanzar cualquier posición que se haya guardado previamente (como *all_zeros* o *home* mostrados en la imagen 16) o se puede mover con el ratón el brazo robótico hasta la posición deseada.

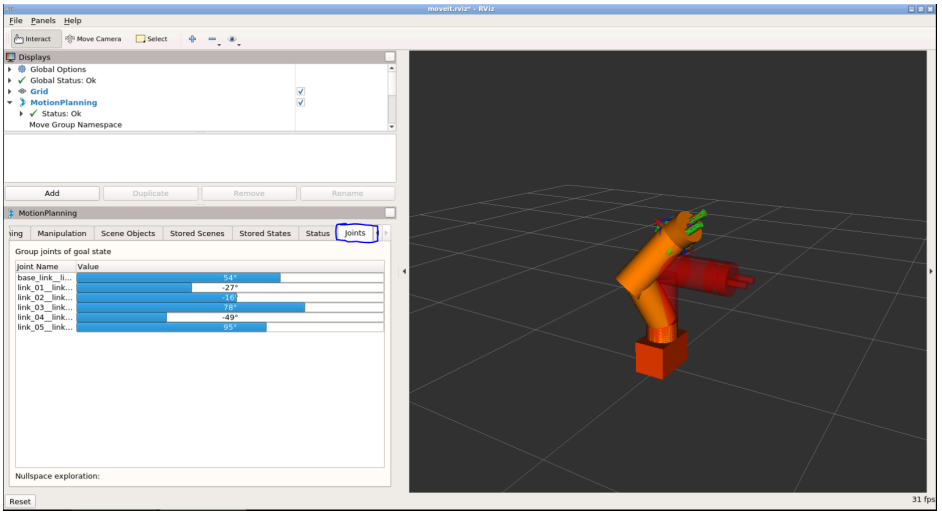

Imagen 26: Panel joints

En la pestaña de *joints* también se puede indicar una posición deseada mediante la posición articular de cada articulación, tal y como se muestra en la imagen anterior.

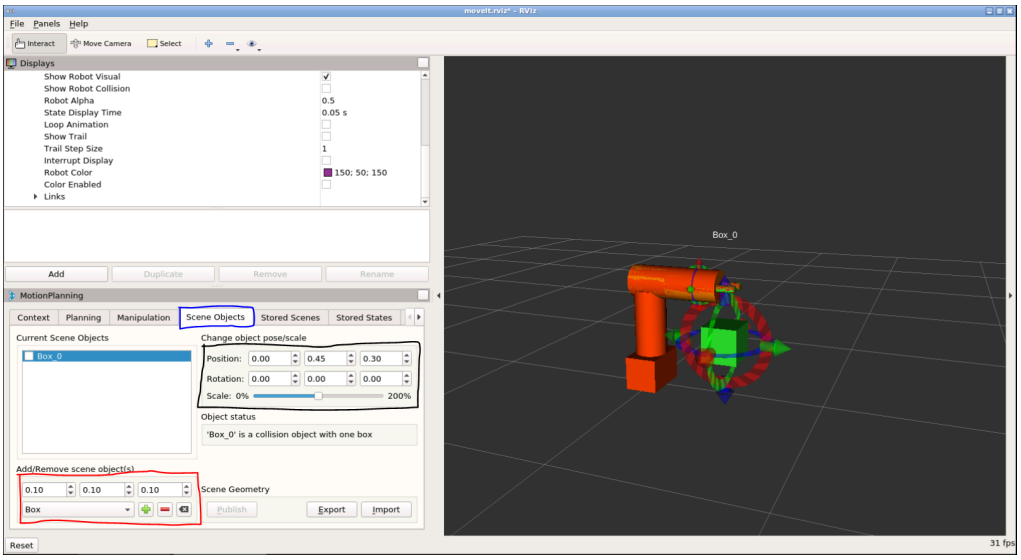

Imagen 27: Panel scene objects

En la pestaña *Scene Objects* se pueden añadir objetos de diferentes formas al escenario creado, en la parte marcada en rojo se puede seleccionar la figura a añadir y su tamaño, mientras que en la zona marcada en negro se puede definir su posición y orientación, también se puede mover con el ratoón el objeto hasta la posición deseada. Además se pueden añadir diferentes objetos, tal y como se muestra en la siguiente imagen.

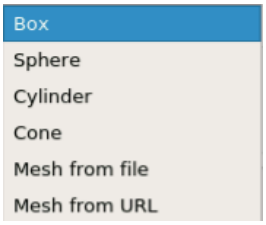

Imagen 28: Objetos que se pueden añadir.

Además estás no son las únicas funcionalidades que tiene el fichero *demo.launch*, si el robot tuviera configurado algún sensor podría detectar objetos, cogerlos y soltarlos como le indiquemos, también se pueden almacenar las escenas y cargarlas cuando quieras.

Conclusiones

Para concluir este tutorial, se ha podido observar que , disponiendo de unos conocimientos medianamente básicos por parte del usuario a cerca de la estructuración y modelado de las partes de un brazo robot, cualquier usuario puede desarrollar el URDF de cualquier robot articulado del cual disponga las características técnicas necesarias y, posteriormente, el diseño y simulación de las trayectorias del brazo robot en un entorno virtual,

Estos aspectos, sumados al hecho de que se trata de un software implementado en ROS pero que cuenta con una interfaz para facilitar la manipulación de diversos parámetros del conjunto, hacen de MoveIt! un programa ideal para introducir al usuario de una manera mucho más sencilla y didáctica a la programación en ROS de este y de una gran variedad de robots disponibles y de la implementación de complejas metodologías aplicables en robots como el mapeado de entornos.

Video

https://drive.google.com/file/d/1hbcFAMPuC6VUxVjp3zT4dBCrIwX4oMux/view?usp=sharing

Referencias

[1] Chitta, S., Sucan, I., & Cousins, S. (2012). Moveit![ros topics]. IEEE Robotics & Automation Magazine, 19(1), 18-19.

[2] ROS Documentation: <link> element
http://wiki.ros.org/urdf/XML/link

[3] ROS Documentation: <joint> element
http://wiki.ros.org/urdf/XML/joint