Universidad Rey Juan Carlos

Práctica 3-B: Integración y estudio de dinámicas en Gazebo y ROS2

Modelado y Simulación de Robots

v 0.3

Objetivo:

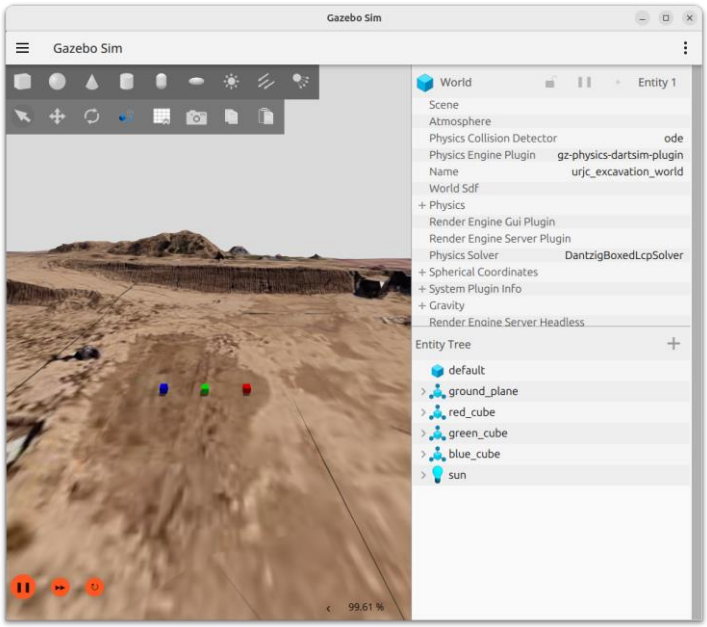
Realización del análisis de coste del mecanismo pick and place (equivalente a la Fase 3 de la práctica 2) en el simulador Gazebo mediante el framework de MoveIt2. Para ello, se utilizará el robot realizado en la práctica 2.

1. Encontrar posiciones deseadas del brazo

Con los archivos generados en la Parte A de esta práctica podemos encontrar las posiciones del brazo y la pinza para poder agarrar la caja. Para ello, usamos los sliders del “joint state" en RViz y la simulación proporcionada.

A partir del mundo de simulación *urjc_excavation_msr* se deberá situar al robot en la posición (0, 0) para coger la caja verde situada en el (5, 0).

https://github.com/juanscelyg/urjc-excavation-world



Este mundo se lanza mediante el comando:

```
ros2 launch urjc_excavation_world urjc_excavation_msr.launch.py
```

Recuerda anotar los puntos de interés obtenidos como posiciones de apertura/cierre de pinza, acercamiento a la caja, posición para agarrar la caja y para dejarla en el robot se deben introducir después como “group states” en el asistente de Moveit.

2. Configuración de MoveIt

Generar un paquete con la configuración del controlador del brazo. Por convención estos paquetes suelen llamarse <nombre robot>_moveit_config. Se recomienda utilizar el asistente de configuración de moveit.

Recuerda hacer las modificaciones necesarias para que los controladores funcionen correctamente, como añadir el fichero <robot name>.srdf al <robot name>.urdf.xacro, revisar el joint_state_limits.yaml, etc. Ver vídeo disponible en Aula Virtual.

3. Launch del robot

A continuación, en el paquete generado en la Parte A de la práctica, se debe añadir un fichero *launch* donde no sólo se lance el mundo, RViz y el robot, si no también el robot *state publisher*. Ese último debe ser el generado por MoveIt disponible en el archivo rsp.launch.py de la configuración anterior. Para poder controlar las ruedas tienes que realizar una transformación del *topic* de velocidad a tipo *stamped*. Ver fichero ejemplo en Aula Virtual.

Ten en cuenta que para que todo funcione correctamente todos los nodos deberán tener activado el parámetro “**use sim time**”.

4. Launch de los controladores

Además, en este paquete del robot, se debe generar un *launch* file que configure y ejecute los controladores. Deberá tener un controlador por cada grupo a mover (base, brazo y pinza) además de lanzar el *broadcaster* del estado de las articulaciones. Ver fichero ejemplo en Aula Virtual.

5. Ejecución

Después de lanzar la simulación y que cargue correctamente, se debe ejecutar el control de movimiento del brazo que se ha generado en el *setup assistant* de MoveIt2. El comando es:

```
ros2 launch <robot name>_moveit_config move_group.launch.py
```

Una vez configurado correctamente, debería aparecer un mensaje que permite comenzar la planificación. En este momento se deben lanzar los controladores del robot:

```
ros2 launch <pkg> robot_controllers.launch.py
```

A partir de este punto se puede trabajar en RViz con el plugin de MoveIt, y se puede teleoperar la plataforma móvil a través del topic `/cmd_vel`.

6. Análisis del mecanismo

El robot deberá situarse en la posición (0, 0) y ser teleoperado hasta una posición que le permita manipular la caja verde que se encuentra situada en el (5, 0). Una vez situado correctamente, deberá coger la caja y depositarla en el robot. Durante todo este proceso se deben almacenar los datos en un rosbag para posteriormente ser analizados.

Se puede mover la plataforma móvil utilizando el paquete teleop_twist_keyboard de la siguiente manera:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

El movimiento del brazo se puede realizar utilizando el plugin *MotionPlanning* de RViz. Para ello, se debe seleccionar el grupo de *joints* que se quiere mover y hacer *click* en el botón *Plan and Execute*.

En esta fase nos interesa estudiar el gasto (G) en términos de potencia que tiene nuestro mecanismo de “pick and place”. Para ello obtendremos las fuerzas aplicadas en cada *joint* del mecanismo “pick and place” (brazo + *end effector*) utilizando la información del esfuerzo en cada *joint*. Se obtendrán medidas desde el momento que el mecanismo inicia el “pick” hasta que vuelve a su posición de origen.

El gasto (G-parcial) en cada momento se define como el sumatorio de las fuerzas aplicadas a cada JOINT involucrado en la cinemática inversa.

$$G = \sum_{i=0}^{N} |F_i|$$

Siendo:

- N, número de joints del mecanismo “pick and place” (involucrados en cinemática inv.)
- F, fuerzas de par motor aplicadas al joint

7. Generación de gráficas

Con los datos obtenidos de la teleoperación se debe generar un **plot “tiempo vs G-parcial”**, donde se muestran el gasto (G-parcial) de cada articulación involucrada en el mecanismo pick and place. Generar un único gráfico con la información de todas las articulaciones necesarias.

Además, se deben generar dos gráficos más con la información obtenida de la teleoperación de la base: **plot “tiempo vs posición de cada una de las ruedas”** obtenida del estado de la articulación y **"tiempo vs aceleración de las ruedas”** obtenido de la IMU.

Se recomienda utilizar herramientas de visualización como pueden ser:

- Plotjuggler: https://github.com/facontidavide/PlotJuggler

En el repositorio anterior se encuentra la información suficiente para poder ejecutar ese paquete en ROS y ver las gráficas deseadas con tan solo dar clicks a los elementos a visualizar.