Universidad Rey Juan Carlos

Práctica 4: Integración de robot en middleware

Modelado y simulación de robots

En esta última práctica se integrará un robot en el simulador Gazebo. Se añadirán diferentes plugins, se teleoperará y se hará una evaluación del recorrido.

Los objetivos para esta práctica son:

- Modificar el robot para hacerlo compatible con SDF.
- Integrar el robot URDF en el simulador Gazebo añadiendo todos los elementos del mundo.
- Añadir plugins de teleoperación y cámara al modelo URDF.
- Evaluación del recorrido del robot URDF.

Uso de ROS y Gazebo

La práctica debe ser probada en los ordenadores de los laboratorios y se recomienda utilizarlos a lo largo de la misma para su desarrollo. A la hora de la evaluación, se corregirán las prácticas según su ejecución en los ordenadores de los laboratorios.

Para ejecutar ROS en los ordenadores:

```
source /opt/ros/noetic/setup.bash
```

Podemos comprobar que podemos ejecutar ROS ejecutando:

```
roslaunch
```

En caso de error devolverá:

```
spaniego@f-l2108-pc09:~$ roslaunch

No se ha encontrado la orden «roslaunch», pero se puede instalar con:

apt install python3-roslaunch
Pregunte al administrador.
```

En caso de que ROS esté disponible devolverá:

```
spaniego@f-l2108-pc09:~$ roslaunch
Usage: roslaunch [options] [package] <filename> [arg_name:=value...]
       roslaunch [options] <filename> [<filename>...] [arg_name:=value...]

If <filename> is a single dash ('-'), launch XML is read from standard input.

roslaunch: error: you must specify at least one input file
```

Lo mismo para comprobar que gazebo está disponible. Podemos ejecutar:

```
gazebo
```

Y debería aparecer el simulador vacío en pantalla.
En los ordenadores de los laboratorios contamos con Ubuntu 20, ROS Noetic y Gazebo 11. Se pueden instalar en local las diferentes versiones siguiendo las guías:

- Instalar ROS Noetic → http://wiki.ros.org/noetic/Installation/Ubuntu
- Instalar Gazebo → http://gazebosim.org/tutorials?tut=install_ubuntu

Fase 1: Modificación del robot a SDF e integración en simulador

En la primera fase de esta práctica, se transformará el diseño del robot que se adjunta en el Aula Virtual en URDF a formato SDF.

Además de eso, se integrará el robot en un escenario vacío con:

- Una rampa de iguales características a la de la práctica anterior, pero en formato SDF. Descripción anterior práctica: *Una estructura que provea una rampa de subida y una de bajada (elemento azul). La altura máxima de este bloque es de 1.5 metros.* Se puede reutilizar el elemento de la práctica anterior, con las adaptaciones que sean necesarias para que el robot completo entre en la rampa y que sea SDF. Se posicionará en (10, 0, 0)
- Un cubo verde (largo_arista=0.5 metros y peso=0.1 kg), que debe ser también SDF y estar en en la posición (20, 0, 0).

Se creará el archivo *.world* y se añadirá el modelo descrito, además del robot, que irá en la posición (0, 0, 2) y se creará un archivo *.launch* para lanzar el mundo.

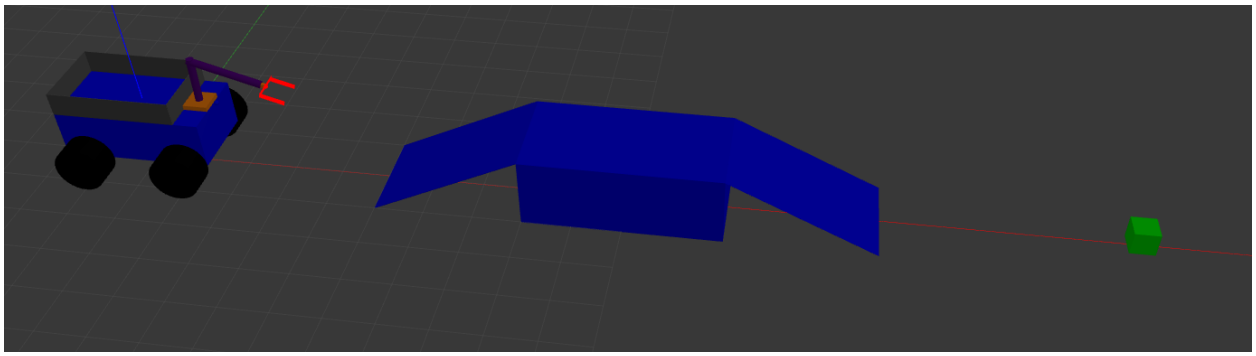
Fase 1b: Integración en simulador del robot URDF

Se integrará el modelo robótico URDF adjunto en el Aula Virtual en la simulación de igual forma que se pide en el punto anterior, pero sin modificar su extensión *.URDF*.

Para este paso se utilizará el mismo mundo creado en la Fase 1a, todo ello con las características previamente descritas.

Se generarán 2 archivos .launch:

- El primero lanzará todo el mundo menos el robot.
- El segundo lanzará el robot en URDF en la posición (0, 0, 2).



Fase 2: Extensión del robot URDF

En la segunda fase, se añadirán al robot URDF varios elementos para dotarlo de una mayor funcionalidad. Se añadirá:

- Una cámara fija en el chasis del robot que permita visualizar lo que ve delante de él.
- Otra cámara al final del brazo mecánico (end-effector), que permita visualizar las piezas a la hora de recogerlas.
- Un controlador para el robot que permita modificar la velocidad a la que se desplaza el chasis joint a joint.
- Un controlador para el brazo mecánico que permita realizar movimientos para recoger el cubo de forma teleoperada.

Las imágenes capturadas por las cámaras serán RGB de 320x240 pixeles y se publicarán en topics diferentes.

Toda esta funcionalidad se proporcionará al usuario mediante un programa teleoperador escrito en python que utilizará *rospy* para este fin. El programa teleoperador contará con una interfaz gráfica (se puede utilizar *PyQt5* para este punto, por ejemplo) que proporcionará las 4 funcionalidades. En la interfaz gráfica se mostrarán las 2 cámaras, además de los 2 teleoperadores (controladores). El diseño de la interfaz es libre.

Así, por ejemplo, el usuario iniciará el programa teleoperador, donde se visualizarán las cámaras y podrá comandar el chasis del robot para que se mueva a lo largo de la superficie o el brazo mecánico.

El alumno **no** debería de tener que implementar código de plugins en C++.

Referencia:

- PyQt5 docs: https://www.riverbankcomputing.com/static/Docs/PyQt5/

Fase 3: Evaluación del recorrido del robot

Una vez se tengan las 2 fases previas completas, para esta última parte de la práctica se querrá teleoperar el robot URDF desde la posición inicial (0, 0, 2) a la posición en la que se encuentra el cubo (20, 0, 0), recogiendo el cubo y volviendo a la posición inicial mientras se evalúa cómo se comporta nuestro robot.

Para ello, se almacenará en un fichero .csv la siguiente información proporcionada por el robot mientras se maneja el robot hasta el cubo, se realiza la recogida del objeto y se vuelve a la posición inicial:

tiempo(seg), posicion_robot[x], velocidad_real_robot, velocidad_comandada

Estos datos se recogerán cada 0,1 segundos. A partir de esos datos se desarrollará un script python que genera una serie de plots para estudiar el comportamiento del robot a lo largo de la simulación.

Se mostrará en una gráfica temporal la evolución de la velocidad real del robot y la velocidad comandada.

Se mostrará en otra gráfica la evaluación de la velocidad real del robot y la velocidad comandada teniendo en cuenta la posición del robot.

Opcional: Filtro de color que detecta la caja

Este punto es opcional, sirviendo como añadido a la calificación final.

Se añadirá a la funcionalidad de la cámara fijada al final del brazo mecánica la posibilidad de detectar la caja dentro de su campo de visión. Para ello, esta cámara procesará las imágenes que le lleguen durante la simulación con un filtro de color (la caja es verde) y avisará al usuario

cuando se detecte la caja. Así, el usuario podrá saber en qué instante tiene la caja dentro de su rango de visión para poder recogerla.

En la parte GUI del programa teleoperador se añadirá otro visualizador para la imagen procesada, teniendo así la visualización de las 2 cámaras sin procesar y la de la imagen procesada.

Se deja a elección del alumno cómo mostrar el aviso de detección, con la única condición de que debe de ser visible dentro de la interfaz gráfica que conforma el teleoperador.

Entrega

La entrega se realizará en fecha y forma definida en el aula virtual. Se generará un fichero comprimido llamado Nombre_Apellido_Practica4.zip con el siguiente contenido.

- Modelos SDF y URDF, además del resto del contenido del paquete, para que sea fácilmente ejecutable a la hora de evaluarlo. Se adjuntará un README.md con instrucciones para ejecutar cada fase.

- Ficheros CSV (campos separados por ,) con las métricas. Debe tener el siguiente formato:

tiempo(seg), posicion_robot[x], velocidad_real_robot, velocidad_comandada

    - Fase3_Nombre_Apellido.csv

- Ficheros PNG/JPG/PDF de los 2 plots generados (fase 3).

    - Fase3_Nombre_Apellido.pdf

- Vídeo autoexplicativo con audio donde contenga al menos los siguientes puntos:

    - Ejecución de la Fase 3 donde se muestran las diferentes cámaras y cómo se teleoperan los controladores.
    - Explicación y razonamiento de los plots.
    - Explicación y razonamiento de las decisiones de diseño y de los controladores.