Escuela Politécnica Superior
Grado en Ingeniería Robótica
Programación de Robots

Universitat d'Alacant
Universidad de Alicante

Práctica 3: Herramienta Transform

Profesor: Antonio Requena Jiménez

Instrucciones

A lo largo de esta práctica vamos aprender qué es la herramienta ***transform(tf)*** de ROS, cuál es su principal utilidad y por qué es tan importante en un entorno robótico.

1. Creación del espacio de trabajo de esta práctica

Para esta práctica vamos a estar trabajando con un entorno ya creado. Para ello descárgate el paquete de ROS turtle_tf_3d que puedes encontrar en el Moodle de la asignatura, añádelo a la carpeta */src* de tu espacio de trabajo. Tambien tienes que copiar la carpeta spawn_robot_tools_pkg y ejecuta el comando *catkin_make* para compilar este paquete.

```
$ chmod +x turtle_tf_3d/scripts/turtle_tf_broadcaster.py
$ chmod +x turtle_tf_3d/scripts/turtle_tf_listener.py
$ chmod +x turtle_tf_3d/scripts/turtle_twist_keyboard.py
$ chmod +x turtle_tf_3d/scr/turtle_tf_3d/get_model_gazebo_pose.py
$ chmod +x turtle_tf_3d/scr/turtle_tf_3d/move_generic_model.py
```

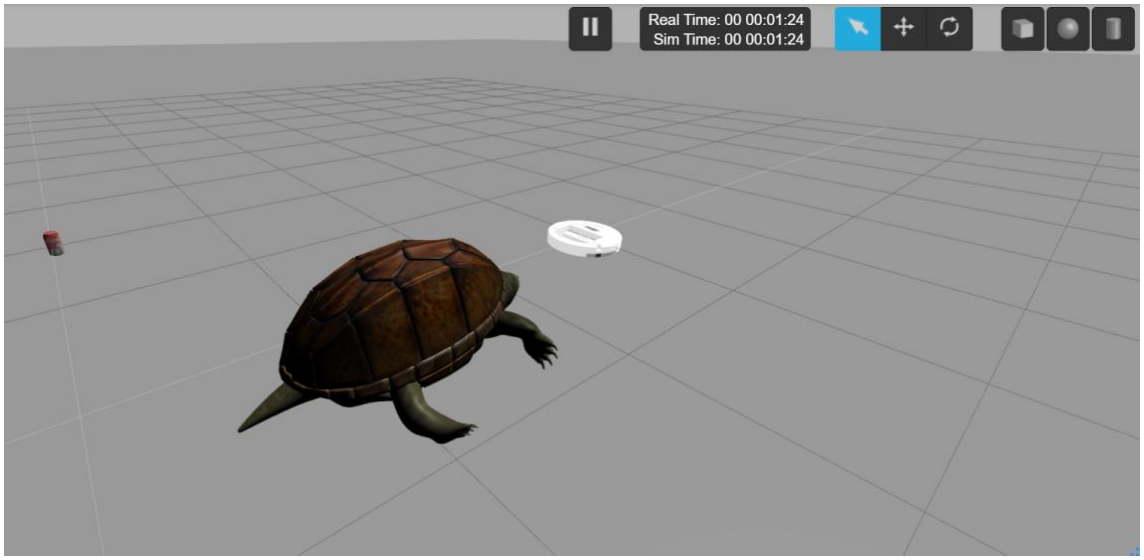
Ahora debes ejecutar los siguientes comandos para dar permisos de ejecución a los ficheros **.py*:

```
$ roslaunch turtle_tf_3d main.launch
```

2. Conociendo el espacio de trabajo

En este apartado vamos a conocer que nos ofrece el espacio de trabajo generado. Para ello ejecuta el siguiente comando:

Este comando ha generado un entorno en Gazebo, como se puede apreciar en la imagen



Si observamos el fichero *main.launch*, podemos ver que lanza el entorno de *gazebo* con el mundo que está almacenado en el fichero *model.world*, que si vemos este archivo tiene un elemento llamado *coke_can* que lo lanza en la posición [*0 2 0*]. Además, lanza tres **.launch* más, uno para poner la tortuga y el robot en el entorno *gazebo*, y los otros dos para darle movilidad a dichos modelos.

Por lo tanto, este entorno nos muestra una tortuga y un robot aspirador, que podremos desplazar por el entorno, y una lata de refresco de cola fija. Esto nos va a servir para conocer como relacionar en ROS los distintos sistemas de referencia que tiene cada elemento, con el fin de que cada uno conozca la posición del otro.

P1. Observa todos los *topics* que se han generado y contesta a las siguientes preguntas, ¿en qué *topic* publicarías para desplazar a la tortuga y al robot aspirador? ¿Qué tipo de mensaje tienes

```
$ roslaunch turtle_tf_3d turtle_keyboard_move.launch
```

que enviar? ¿Hay algún nodo publicando o suscrito a dichos *topics*?

Para poder mover la tortuga se ha creado un *launch* que detecta las pulsaciones del teclado y realiza el movimiento de la tortuga acorde a la tecla pulsada. Para ello ejecuta el siguiente comando:

Una vez lanzado, teniendo el cursor en la terminal donde se ha lanzado este *launch*, podemos mover el robot hacia delante y rotarlo a derecha e izquierda mediante las teclas *[i, o, u]*, respectivamente. Para detener el movimiento pulsaremos la tecla *[espacio]*. Estas son las principales teclas que vamos a usar, pero también existen distintos movimientos asociados a otras teclas.

P2. ¿Qué nodo ha lanzado este launch y a qué *topic* se ha suscrito? ¿Hay ahora algún nodo publicando en el alguno de los *topics* de la P1?

Ahora que ya conocemos un poco el entorno que hemos generado, vamos a conocer la utilidad de la herramienta ***transform***.

3. Herramienta Transform

En un entorno robótico suelen haber multitud de sistemas de referencia, como por ejemplo los de un brazo robótico, el de una cámara o el de un sensor láser. Además, estos sistemas de referencia pueden variar si posición a lo largo del tiempo. Por lo tanto, debemos conocer la relación que existe entre los diferentes sistemas de referencia para poder trabajar en un entorno robótico.

Esta necesidad es abordada en ROS mediante la herramienta ***transform***, y vamos a aprender cómo funciona poniendo como ejemplo el entorno generado anteriormente. Ahora

```
$ roslaunch turtle_tf_3d  irobot_follow_turtle.launch
```

vamos a lanzar el siguiente comando:

Podemos observar como el robot aspirador quiere alcanzar la posición de la tortuga. Esto ha sido posible gracias al ***tf***, que ha relacionado la transformación (*frame)* de la tortuga y la transformación (*frame*) del robot. Lanza el *launch* turtle_keyboard_move para mover la tortuga y ver como el robot intenta seguir la trayectoria de la tortuga.

P3. ¿Qué nodos se han lanzado a través de este **.launch* y qué nuevos *topics* han aparecido?

```
$ rosrun tf view_frames                                    # Genera un pdf
$ rosrun rqt_tf_tree rqt_tf_tree
$ rostopic echo –n2 /tf
```

El *topic /tf* es el */topic* por el que se están publicando los *frames* de la tortuga y del robot. Para ver la relación entre las transformaciones de estos dos *frames* podemos utilizar los siguientes comandos:

P4. ¿Cómo se llaman los dos *frames*? ¿Con que *transform* están relacionado ambos? ¿Cuál es el *frame* 'padre' y el 'hijo'?

```
$ rosrun tf tf_echo turtle1 turtle2
```

Mediante estos comandos hemos visto las relaciones existentes de los dos *frames* creados a partir del **.launch* con respecto al *frame world*, el 'padre'. Sin embargo, para que el robot aspirador pueda seguir a la tortuga, lo que debe conocer el robot es la relación existente entre su *frame* y el de la tortuga. Para conocer esta transformación basta con ejecutar el comando:

De este modo, somos capaces de conocer la relación existente entre estos dos elementos,

```
$ roslaunch turtle_tf_3d main.launch
$ roslaunch turtle_tf_3d  irobot_follow_turtle.launch
```

donde turtle1 es la tortuga y turtle2 el robot.

```
$ roslaunch turtle_tf_3d turtle_keyboard_move.launch
```

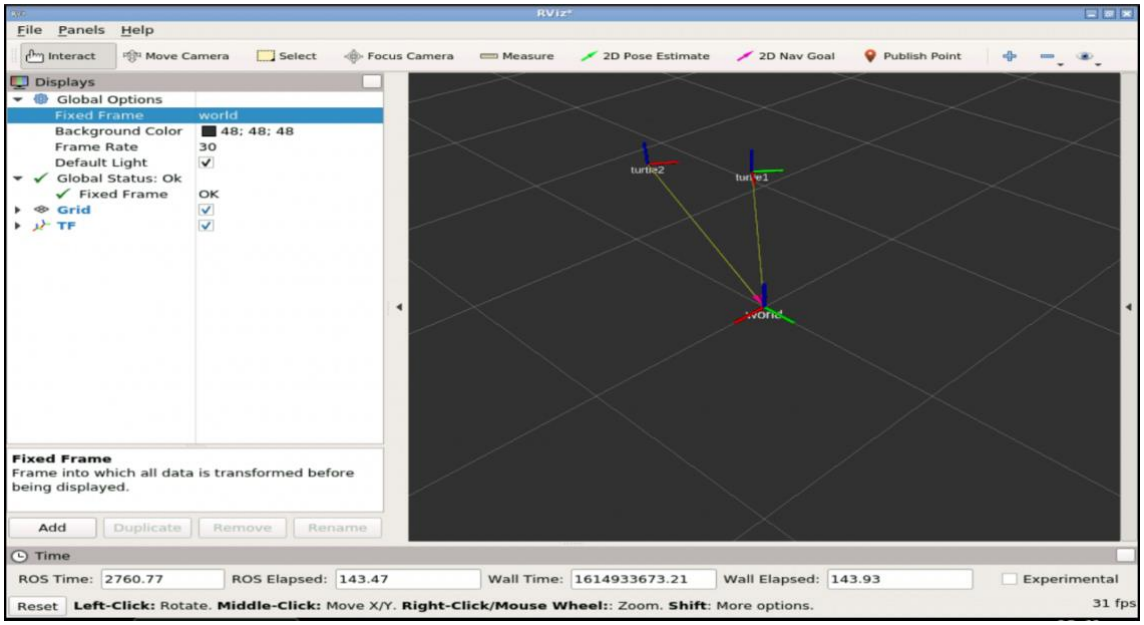
Por último, vamos a utilizar *rviz* para observar visualmente el comportamiento de la herramienta *transform*. Para que no haya problemas, vamos a parar la simulación por completo, parando todos los nodos que hayamos lanzado en las diferentes terminales. Ahora ejecutamos:

A continuación, vamos a dejar a la tortuga moviéndose en círculos pulsando la tecla [*o*] tras ejecutar lo siguiente:

Una vez que la tortuga se esté moviendo, vamos a lanzar rviz ejecutando:

```
$ rosrun rviz rviz
```

Ahora ponemos como *Fixed Frame* de rviz el *tranform world* y añadimos la herramienta *transform* a nuestro entorno *rviz* quedando algo parecido como lo que se muestra en la siguiente imagen.



Como se puede observar, estamos representando los *frames* de la tortuga y el robot con respecto al *frame* mundo.

P5. ¿Qué sucede si cambiamos nuestro *Fixed Frame* por otro *frame*? ¿Hacia dónde apuntan las flechas amarillas y qué indica esto?

Como conclusión de este apartado podemos decir que la herramienta *transform* nos permite relacionar todos nuestros sistemas de referencia, fijos y móviles, en un mismo *topic*, para poder conocer las relaciones de todos los sistemas que tiene nuestro entorno robótico.

Ejercicios

1. Abre el fichero *turtle_tf_listener.py* observa que es lo que hace este código y detecta la línea por la que se obtiene la transformación entre dos *frames*. ¿Qué tipo de objeto es y que función se usa para este fin?
   Ahora abre el fichero *irobot_follow_turtle.launch* y *turtle_keyboard_move.launch* y estudia los nodos que lanza.
   - Crea un fichero *turtle_follow_irobot.launch* para que la tortuga sea la que sigue al robot.
   - Crea un fichero *irobot_keyboard_move.launch* para poder mover el robot mediante las pulsaciones de teclado.

2. Crea un *script* llamado *turtles_and_coke_broadcaster.py* que sea capaz de publicar los *frames* de las dos tortugas y del refresco de cola. Recuerda que el modelo del refresco de cola se llama *coke_can*.
3. Crea un *script* llamado *turtles_and_coke_listener.py* que muestre por la terminal diferencia en la posición x y en orientación y.
4. Haz un fichero llamado *turtles_and_coke.launch* que lance los dos nodos creados en los ejercicios 2 y 3.