Memoria 30.03.22
Práctica 2 - Mostrar en RVIZ múltiples robots.

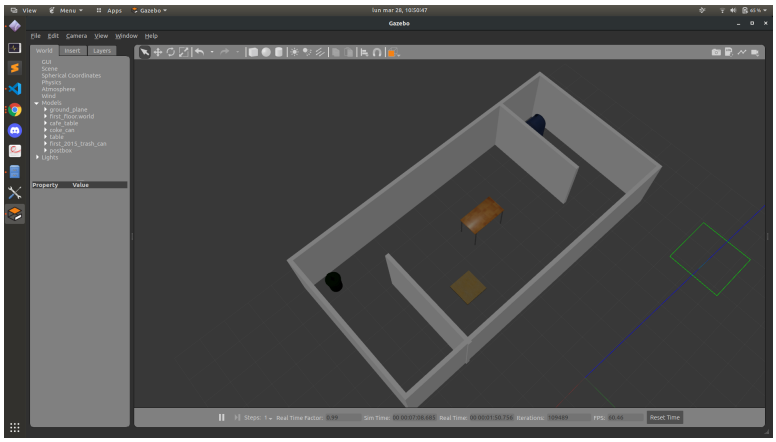
Ignacio Pérez Vilaplana

1. Introducción

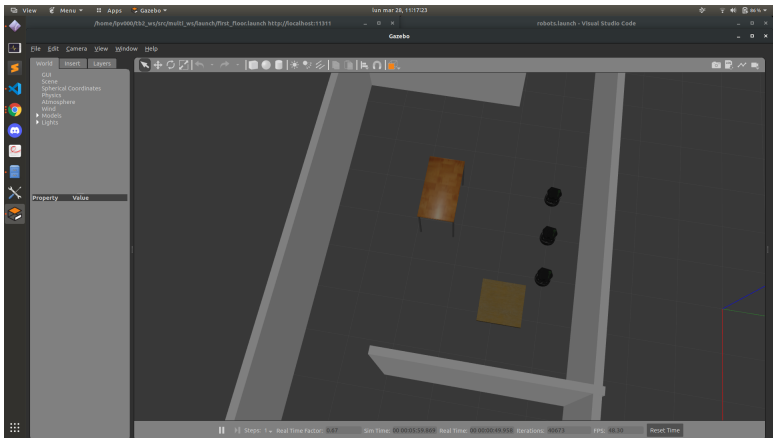
Esta práctica se entrega habiendo empezado ya la tercera bajo la justificación de que, aún estando hecha desde hace más de una semana no se ha podido hacer la entrega. Esto es debido a que se han ido utilizando las entregas abiertas para subir las memorias de las sesiones en donde hemos tenido trabajo con el robot real y no con simulación.

2. Creación de un nuevo mundo en Gazebo que contenga múltiples objetos

Tal y como se pide en la guía de la práctica, lo primero, es crear un nuevo mundo que contenga diversos objetos. Se ha creado una planta de 4 paredes, con dos separaciones y 4 objetos diferentes: un buzón de correos, dos mesas y una papelera.

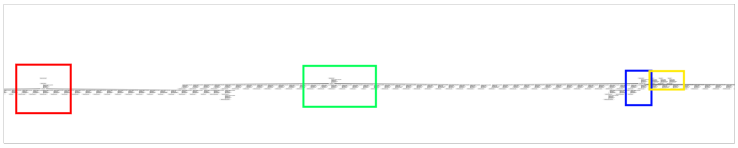


Cargamos los 3 robots en el nuevo mundo y comprobamos que están dentro de las 4 paredes.



3. Visualización de la estructura de los nodos de los robots con *rqt_tf_tree* y muestra en RVIZ

Antes de hacer ningún cambio en código, ni de añadir o quitar ninguna línea con respecto a los códigos de la práctica anterior, visualicemos como se tiene el árbol de transformaciones inicialmente.



El árbol es enorme y es por ello que se han marcado 4 áreas. En rojo, están los nodos del robot2/base_footprint, en verde los del robot3/base_footprint y en azul los del robot1/base_footprint. Por otro lado en el cuadro amarillo tenemos 3 nodos separados que corresponden a robot1/odom, robot2/odom y robot3/odom.

A partir de aquí, ¿qué se puede hacer para juntarlo todo bajo el mismo marco de transformaciones? Se tiene que el robot1 y el mapa o entorno creado si dependen el primero del segundo, sin embargo los otros dos no. Esto hace que robot2 y

robot3 no pertenezcan al mismo marco global de transformaciones. Para que todo este bajo el mismo marco de coordenadas se va a emplear el paquete de transformaciones *tf* de ROS.

TF permite realizar un seguimiento de todos los sistemas del entorno a lo largo del tiempo y nos permite hacer y resolver preguntas como: cuál es la cabeza del marco global, o que transformaciones tiene cada uno de los sistemas con respecto al general. De esta forma para conseguir nuestro objetivo necesitamos que cada uno de nuestros sistemas, en este caso, robot1, robot2 y robot3, envíen información al marco general.

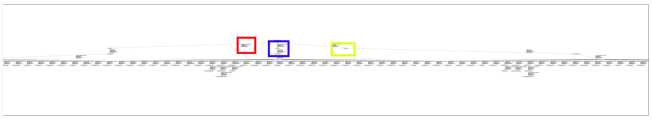
El archivo que se ha modificado para que todo en el mismo marco es el de *robots.launch*. Este contiene un código repetido (pero con diferentes parámetros de posición inicial) para los 3 robots. El código es el siguiente:

```
4           <!-- BEGIN ROBOT 1-->
5               <group ns="robot1">
6                   <param name="tf_prefix" value="robot1_tf" />
7                   <node pkg="tf"
                    ↪  type="static_transform_publisher"
                    ↪  name="robot1_broadcaster" args="0 -4 0 0 0
                    ↪  3 map robot1_tf/odom 100" />
8                   <include file="$(find
                    ↪  multi_ws)/launch/one_robot.launch" >
9                       <arg name="init_pose" value="-x 0 -y -4 -z
                        ↪  0 -R 0 -P 0 -Y 3" />
10                      <arg name="robot_name" value="Robot1" />
11                  </include>
12              </group>
```

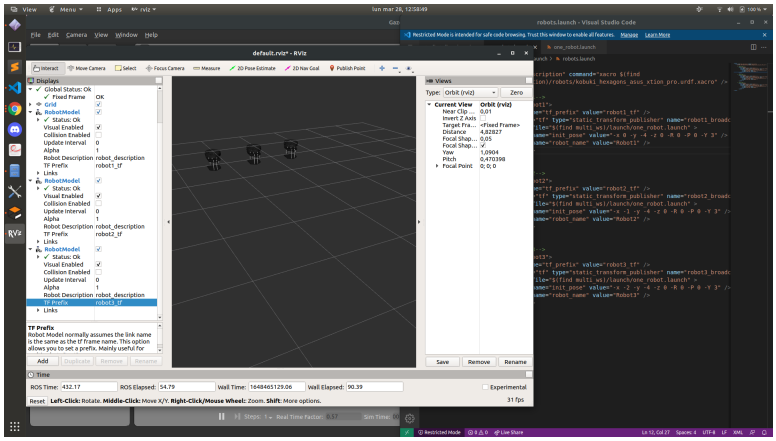
La línea que se ha añadido es la número 7. Al añadir este paquete y crear el nodo emisor (broadcaster) lo que se hace es enviar la información del robot al marco global, *map*.

```
<node pkg="tf" type="static_transform_publisher"
↪  name="robot1_broadcaster" args="0 -4 0 0 0 3 map
↪  robot1_tf/odom 100" />
```

Es de esta forma como se consigue una correcta representación en el árbol de transformaciones *tf*. El nuevo arbol puede verse a continuación. Si se amplia la foto puede verse que existe unión entre los sistemas de los robots y el marco global *map*.



De esta forma se pueden ver los robots representados en RVIZ de la siguiente forma. Para mostrarlos se ha tenido que añadir 3 tipos *RobotModel* e incluir el mismo prefijo que se les ha dado en su declaración al lanzar el nodo en el fichero *.launch*. En cuanto se han añadido se han mostrado.



4. Topics obtenidos y añadidos

Una vez se tiene todo en marcha se hace un

```
>rostopic list
```

para mostrar todos los topics que existen. Para los 3 robots son los mismos. Para que no extender mucho la memoria se van a copiar solo los del robot1.

```
/robot1/camera/depth/camera_info
/robot1/camera/depth/image_raw
/robot1/camera/depth/points
/robot1/camera/parameter_descriptions
/robot1/camera/parameter_updates
/robot1/camera/rgb/camera_info
/robot1/camera/rgb/image_raw
/robot1/camera/rgb/image_raw/compressed
/robot1/camera/rgb/image_raw/compressed/parameter_descriptions
/robot1/camera/rgb/image_raw/compressed/parameter_updates
/robot1/camera/rgb/image_raw/compressedDepth
/robot1/camera/rgb/image_raw/compressedDepth/parameter_descript
↪  ions
/robot1/camera/rgb/image_raw/compressedDepth/parameter_updates
/robot1/camera/rgb/image_raw/theora
/robot1/camera/rgb/image_raw/theora/parameter_descriptions
/robot1/camera/rgb/image_raw/theora/parameter_updates
/robot1/depthimage_to_laserscan/parameter_descriptions
/robot1/depthimage_to_laserscan/parameter_updates
/robot1/joint_states
/robot1/laserscan_nodelet_manager/bond
/robot1/mobile_base/commands/motor_power
/robot1/mobile_base/commands/reset_odometry
/robot1/mobile_base/commands/velocity
/robot1/mobile_base/events/bumper
/robot1/mobile_base/events/cliff
/robot1/mobile_base/sensors/core
/robot1/mobile_base/sensors/imu_data
/robot1/odom
```

Al imprimir esto nos hemos dado cuenta de que falta un topic importante, el */scan*. El *LaserScan* se supone que se debería de lanzar teniendo en cuenta nuestro fichero *one_robot.launch*, sin embargo, esto no es así. De esta forma, se procede a editar el fichero para que funcione el nodo correctamente. El fichero, tras realizar los cambios queda tal que:

```
<launch>
    <arg name="robot_name"/>
    <arg name="init_pose"/>
    <node name="spawn_minibot_model" pkg="gazebo_ros"
    ↪  type="spawn_model"
        args="$(arg init_pose) -urdf -param /robot_description
        ↪  -model $(arg robot_name)"
        respawn="false" output="screen" />
```

```
    <node pkg="nodelet" type="nodelet"
    ↪  name="laserscan_nodelet_manager" args="manager"/>
    <node pkg="nodelet" type="nodelet"
    ↪  name="depthimage_to_laserscan" args="load
    ↪  depthimage_to_laserscan/DepthImageToLaserScanNodelet
    ↪  laserscan_nodelet_manager">
        <param name="scan_height" value="10"/>
        <param name="output_frame_id" value="$(arg
        ↪  robot_name)/camera_depth_frame"/>
        <param name="range_min" value="0.45"/>
        <remap from="image" to="camera/depth/image_raw"/>
        <remap from="scan" to="scan"/>
    </node>
    <node pkg="robot_state_publisher"
    ↪  type="robot_state_publisher"
        name="robot_state_publisher" output="screen" >
        <param name="publish_frecuency" type="double"
        ↪  value="30.0"/>
    </node>
</launch>
```

Tras hacer este cambio nuestra lista de topics queda tal que:

```
.
.
/robot1/mobile_base/sensors/core
/robot1/mobile_base/sensors/imu_data
/robot1/odom
/robot1/scan
```

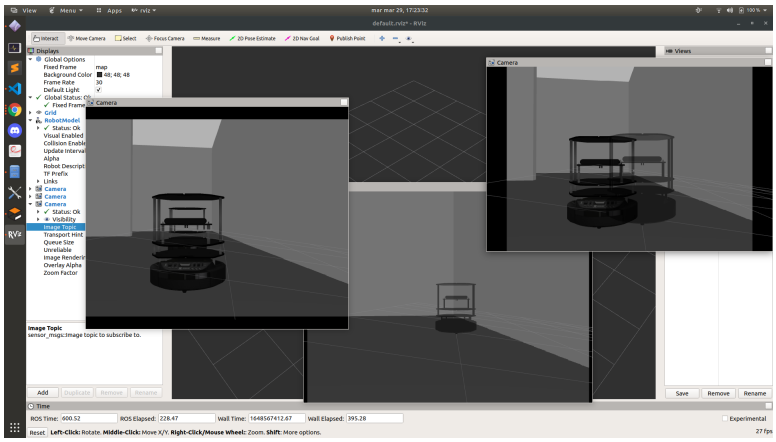
Se puede comprobar como ahora si que aparece el topic */scan*. Para comprobar que las medidas no son idénticas entre los 3 robots se ha copiado el mensaje enviado por el topic en un fichero *.txt*. Esto se ha hecho para cada uno de los robots. Los ficheros tienen el nombre *laserScanR(número_robot)*.

```
> rostopic echo -n 1 /robot1/scan > laserScanR1.txt
> rostopic echo -n 1 /robot2/scan > laserScanR2.txt
> rostopic echo -n 1 /robot3/scan > laserScanR3.txt
```

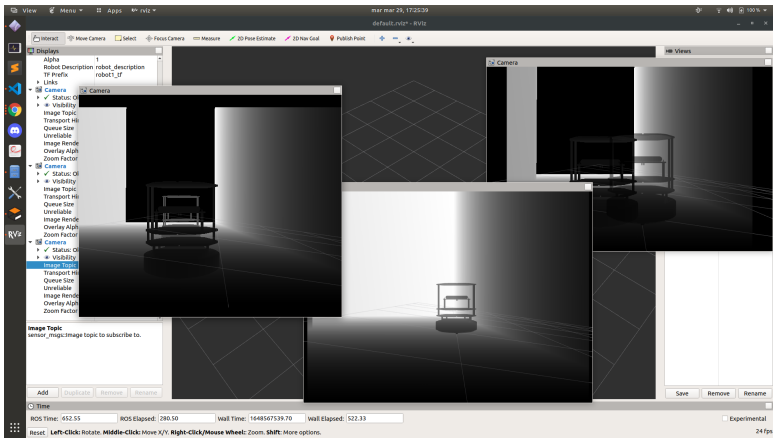
Ahora bien, para hacer la comparación entre cada uno de estos se utiliza el comando de terminal *diff -q -w -B*. Este comando nos indicará si los dos ficheros son distintos o iguales, sin tener en cuenta espacios en blanco o tabulaciones.

```
> diff -q -w -B laserScanR1.txt laserScanR2.txt
"Los archivos laserScanR1.txt y laserScanR2.txt son distintos"
```

Otros dos *topics* que se pueden ver son los de las cámaras. Por un lado tenemos *camera/depth/image_draw* que se puede ver desde RVIZ:



Y por otro lado *camera/rgb/image_draw*, también visible desde RVIZ:



5. Consultas y tutoriales web utilizados

A continuación, los enlaces que se han empleado para realizar la práctica.

http://wiki.ros.org/tf/Tutorials/Introduction%20to%20tf
http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28Python%29
http://wiki.ros.org/tf
http://wiki.ros.org/tf/Overview/Broadcasting%20Transforms