GUÍA

JORGE MUÑOZ RODRÍGUEZ
PERCEPCIÓN Y CONTROL

Índice

Crear Entorno de Desarrollo

1º. Crear workspace

1.1- Crear directorio nombre_ws

1.2- Crear directorio src dentro de nombre_ws

1.3- En nombre_ws, ejecutar catkin init

2º. Crear un paquete

2.1- Entrar al directorio WS/src

2.2- catkin_create_pkg [nombrePaquete] rospy

3º. Crear un nodo en un paquete

3.1.1- En PAQUETE/src, creamos un nuevo archivo nombre.py

3.1.2- Hacer el nodo ejecutable con "sudo chmod +x nombre.py"

3.2- En PAQUETE, crear un directorio launch

3.3- En PAQUETE/launch, crear un archivo nombre.launch

4º. Compilar

4.1- En WS, ejecutar catkin_make

* Una vez compilado, recomendable ejecutar source devel/setup.bash para indicar en que WS estás trabajando.

5º. Para muchos de los comandos siguientes, es necesario tener ejecutándose roscore.

ROSTOPIC

rostopic pub [tópico] [tipoMensaje] [mensaje]
(Publicar en un tópico)
rostopic list
(Listar los tópicos)
rostopic info [tópico]
(Muestra información sobre un tópico)
rostopic list
(Listar los tópicos)
rostopic list
(Listar los tópicos)
rostopic echo [tópico]
(Muestra la salida del tópico)
rostopic info [tópico]
(Muestra la información sobre el tópico)

ROSLAUNCH

roslaunch [rutaNodo] [rutaLaunch]
(Ejecutar un nodo)

ROS/Linux

roscd [rutaNodo]
(Cambiar a un directorio ROS, dentro del WS al que hicimos source devel/setup.bash)

ROSPARAM

rosparam list
(Muestra una lista con las configuraciones del entorno)
rosparam get [parámetro]
(Muestra el valor de un parámetro)
rosparam set [parámetro]
(Modifica el valor de un parámetro)

ROSNODE

rosnode list
(Muestra una lista con los nodos ACTIVOS)
rosnode info [nodo]
(Muestra información sobre un nodo)

TF2

TF2 es la segunda generación de la biblioteca Transform. Permite realizar un seguimiento de varios fotogramas de coordenadas a lo largo del tiempo.

- Si queremos usar TF2, al crear un nuevo paquete:

```
catkin_create_pkg paquete_tf2 tf2 tf2_ros rospy
```

Emisora Estática TF2 (Broadcaster)

- El paquete tf2_ros proporciona un StaticTransformBroadcaster para facilitar la publicación de transformaciones estáticas. Ejemplo de creación de objeto StaticTransformBroadcaster:

```
broadcaster = tf2_ros.StaticTransformBroadcaster()
static_transformStamped = geometry_msgs.msg.TransformStamped()

static_transformStamped.header.stamp = rospy.Time.now()
static_transformStamped.header.frame_id = "world"
static_transformStamped.child_frame_id = sys.argv[1]
```

Requisitos para la creación:

a) Necesitamos darle a la transformación que se está publicando una marca de tiempo. La marcaremos con la hora actual, *rospy.Time.now*.
b) Debemos establecer el nombre del *frame padre* que estamos creando. En este caso, "world".
c) Debemos establecer el nombre del *frame hijo* que estamos creando. En este caso, lo obtenemos de sys.argv[1].

- El paquete tf2_ros proporciona un TransformStamped, que será el mensaje que enviemos una vez rellenado. Ejemplo de rellenado del mensaje:

```
static_transformStamped.transform.translation.x = float(sys.argv[2])
static_transformStamped.transform.translation.y = float(sys.argv[3])
static_transformStamped.transform.translation.z = float(sys.argv[4])

quat = tf.transformations.quaternion_from_euler(
       float(sys.argv[5]),float(sys.argv[6]),float(sys.argv[7]))
static_transformStamped.transform.rotation.x = quat[0]
static_transformStamped.transform.rotation.y = quat[1]
static_transformStamped.transform.rotation.z = quat[2]
static_transformStamped.transform.rotation.w = quat[3]
```

- Finalmente, tenemos que enviar la transformación, de esta forma:

```
broadcaster.sendTransform(static_transformStamped)
```

- Gracias a la nueva versión de TF2, podemos publicar transformaciones como una línea de comandos, en lugar de escribirlo en el código. Se puede hacer de la siguiente forma:

```
static_transform_publisher x y z Pitch Roll frame_id child_frame_id
static_transform_publisher x y z qx qy qz qw frame_id child_frame_id
```

Emisora TF2 (Broadcaster)

- El paquete tf2_ros proporciona también un TransformBroadcaster para facilitar la publicación de transformaciones no estáticas. El funcionamiento es igual al explicado en *Emisora Estática TF2*.

Oyente TF2 (Listener)

- El paquete tf2_ros proporciona un TransformListener para facilitar la tarea de recibir transformaciones. Ejemplo de creación de objeto TransformListener:

```
tfBuffer = tf2_ros.Buffer()
listener = tf2_ros.TransformListener(tfBuffer)
```

- Una vez creado el oyente, comienza a recibir transformaciones a través del *Buffer*, y las almacena un máximo de 10 segundos.

- Un ejemplo de consulta de un oyente para una transformación específica:

```
transformation = tfBuffer.lookup_transform(turtle_name, 'turtle1', rospy.Time())
```

Esto significa:

1) Queremos la transformación de este marco... *(turtle_name)*
2) ... a este marco. *('turtle1')*
3) El momento en el que queremos transformarnos. En este caso, proporcionado por rospy.Time(), esto nos dará la última transformación disponible.

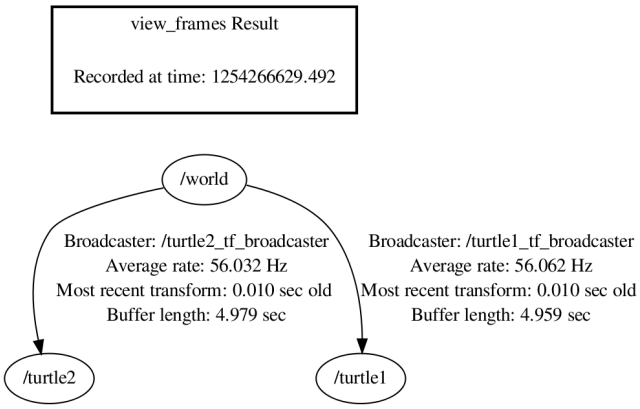
Debe envolverse en un bloque try-except, ya que es frecuente atrapar excepciones.

Crear Marcos TF2 (Frames)

- Para muchas tareas es más fácil pensar dentro de un marco local. TF2 permite definir un marco local para cada sensor, cono, etc. TF2 se encargará de todas las transformaciones de marco adicionales que se introduzcan.

- TF2 construye un árbol de frames. **No permite un bucle cerrado en la estructura del frame**. Es decir, un *frame* tiene un solo *padre*, pero puede tener varios *hijos*.
Por ejemplo:

· Cada círculo es un *frame*.
· El *frame padre* es /world, /turtle1 y /turtle2 son sus *frames hijos*.
· Si queremos agregar un *frame* nuevo, uno de los *frames actuales* debe ser su padre. El nuevo se convertirá en su hijo.



Ejemplo de creación de frame:

```
while not rospy.is_shutdown():
            rospy.sleep(0.1)                          (Frecuencia del bucle)

            t = geometry_msgs.msg.TransformStamped()
            t.header.frame_id = "turtle1"             (Nombre frame padre)
            t.header.stamp = rospy.Time.now()
            t.child_frame_id = "carrot1"              (Nombre frame hijo)
            t.transform.translation.x = 0.0
            t.transform.translation.y = 2.0
            t.transform.translation.z = 0.0
```

-El marco /carrot1 está a 1m en el eje y del frame padre /turtle1.