TF Tutorial

① Introduction to TF

`rosrun tf view_frames`

→ Creates a diagram of the frames being broadcast by tf over ROS.

`rosrun rqt_tf_tree rqt_tf_tree`

`rosrun tf tf_echo [reference_frame] [target_frame]`

→ Reports the transformation between any two frame broadcast over ROS.

→ Translation

Rotation in Quaternion

in Roll Pitch Yaw

① Writing a tf broadcaster (C++)

`#include <tf/transform_broadcaster.h>`

⇒ The tf package provides an implementation of a TransformBroadcaster to help make the task of publishing transforms easier.

⇒ To use the TransformBroadcaster, we need to include the tf/transform_broadcaster.h

br.sendTransform(tf::StampedTransform(transform, ros::Time::now(), "world", turtle_name));

Name of the parent frame

Name of the child frame

② Writing a tf listener (C++)

```
#include <tf/transform_listener.h>
```

→ The tf package provides an implementation of a TransformListener to help make the task of receiving transforms easier.

→ To use the TransformListener, we need to include the tf/transform_listener.h header file.

```
tf::TransformListener listener;
```

→ Here we create TransformListener object.

→ Once the listener is created, it starts receiving tf transformations over the wire, and buffers them for up to 10 sec.

try {

listener.lookupTransform("/turtle2", "/turtle1",
ros::Time(0), transform);

}

→ We want the transform from frame /turtle1 to /turtle2.

→ The time at which we want to transform. Providing ros::Time(0) will get us the latest available transform.

→ The object in which we want to store the resulting transform.

③ Adding a frame (C++)

* Why adding frames

For many tasks it is easier to think inside a local frame.

↓

eg: It is easier to reason about a laser scan in a frame at the center of the laser scanner.

* Where to add frames

⇒ tf builds up a tree structure of frames. it does not allow a closed loop in the frame structure.

↳ Frame only has one single parent, but it can have multiple childres.

* How to add a frame

```
transform.setOrigin(tf::Vector3(0.0, 2.0, 0.0));
transform.setOrigin(tf::Quaternion(0,0,0,1));
br.sendTransform(tf::StampedTransform
(transform, ros::Time::now(), "turtle1", "carrot1"));
```

Parent ⇒ turtle1

child ⇒ carrot1

(4) Learning about tf and time (C++)

* tf and Time

⇒ tf keeps track of a tree of coordinate frames.

↳ This tree changes over time, and tf stores a time snapshot for every transform (for up to 10 sec by default)

*Wait for transform

```
listener.waitForTransform("/turtle2", "/turtle1", now
                          , ros::Duration(3.0));
```

waitForTransform() will actually block until the transform between the two turtle become available until the timeout has been reached.

⑤ Time travel with tf (c++)

try{

```
ros::Time now = ros::Time::now();
ros::Time past = now - ros::Duration(5.0);
listener.waitForTransform("/turtle2", now,
                          "/turtle1", past
                          ,"/world", ros::Duration(1.0));

listener.lookupTransform("/turtle2", now,
                         "/turtle1", past,
                         "/world", transform)
```