ROS tf System

13-15 February, 2017

Janyl Jumadinova
What is tf?

- A robotic system typically has many coordinate frames that change over time, such as a world frame, base frame, gripper frame, head frame, etc.
- tf is a transformation system that allows making computations in one frame and then transforming them to another at any desired point in time.
What is \textit{tf}?

- A robotic system typically has many coordinate frames that change over time, such as a world frame, base frame, gripper frame, head frame, etc.

- \textit{tf} is a transformation system that allows making computations in one frame and then transforming them to another at any desired point in time.

- \textit{tf} allows you to ask questions like:
  - What is the current pose of the base frame of the robot in the map frame?
  - What is the pose of the object in my gripper relative to my base?
  - Where was the head frame relative to the world frame, 5 seconds ago?
Benefits of tf

- Distributed system - no single point of failure.
- No data loss when transforming multiple times.
- No computational cost of intermediate data transformations between coordinate frames.
- The user does not need to worry about which frame their data started.
- Information about past locations is also stored and accessible (after local recording was started).
tf Nodes

There are two types of tf nodes:

- **Publishers** – publish transforms between coordinate frames on `/tf`.
- **Listeners** – listen to `/tf` and cache all data heard up to cache limit.
TF builds a tree of transforms between frames

```
/base_link
  ▶ broadcaster: /base_to_head
      average rate: 10.190 Hz
      most recent transform: -0.027 sec old
      buffer length: 4.907 sec
  ▶ broadcaster: /head_frame
      broadcaster: /HeadMotorHandler
      average rate: 50.201 Hz
      most recent transform: 0.015 sec old
      buffer length: 4.960 sec
  /head_frame
  ▶ broadcaster: /head_to_arm
      average rate: 10.196 Hz
      most recent transform: -0.014 sec old
      buffer length: 4.866 sec
  ▶ broadcaster: /kinect_normal_axis_to_kinect
      average rate: 10.190 Hz
      most recent transform: -0.072 sec old
      buffer length: 4.907 sec
  /kinect_normal_axis_frame
  /kinect_frame
  /arm_frame
```
Transform Tree

TF builds a tree of transforms between frames

- Can support multiple disconnected trees.
- Transforms only work within the same tree.
Communications medium has significant impact:

- **view_frames**: visualizes the full tree of coordinate transforms.
tf Command-line Tools

Communications medium has significant impact:

- **view_frames**: visualizes the full tree of coordinate transforms.
- **tf_monitor**: monitors transforms between frames.
tf Command-line Tools

<table>
<thead>
<tr>
<th>Communications medium has significant impact:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ <strong>view_frames</strong>: visualizes the full tree of coordinate transforms.</td>
</tr>
<tr>
<td>▶ <strong>tf_monitor</strong>: monitors transforms between frames.</td>
</tr>
<tr>
<td>▶ <strong>tf_echo</strong>: prints specified transform to screen.</td>
</tr>
</tbody>
</table>
tf Command-line Tools

Communications medium has significant impact:

- **view_frames**: visualizes the full tree of coordinate transforms.
- **tf_monitor**: monitors transforms between frames.
- **tf_echo**: prints specified transform to screen.
- **roswtf**: with the tfwtf plugin, helps you track down problems with tf.
Communications medium has significant impact:

- **view_frames**: visualizes the full tree of coordinate transforms.
- **tf_monitor**: monitors transforms between frames.
- **tf_echo**: prints specified transform to screen.
- **roswtf**: with the tfwtf plugin, helps you track down problems with tf.
- **static_transform_publisher** is a command line tool for sending static transforms.
Step 1: tf Demo

Follow the tutorial on Introduction to tf ROS Wiki Link

```bash
roslaunch turtle_tf turtle_tf_demo.launch
rosrun turtlesim turtle_teleop_key

rosrun tf view_frames
evince frames.pdf

rosrun rqt_tf_tree rqt_tf_tree
rosrun tf tf_echo turtle1 turtle2

rosrun rviz rviz -d 'rospack find turtle_tf'/rviz/turtle_rviz.rviz
```
Broadcasting Transforms

- A tf broadcaster sends out the relative pose of coordinate frames to the rest of the system.
- A system can have many broadcasters, each provides information about a different part of the robot.
- We will now write the code to reproduce the tf demo.
Step 2: tf Demo

Follow the tutorial on Writing a tf broadcaster

```
  cd ~/catkin_ws/src
  catkin_create_pkg learning_tf tf roscpp rospyn turtlesim
  cd ..
  catkin_make
  roscd learning_tf
```

Make sure to have: `src/turtle_tf_broadcaster.cpp` (copy the program from the shared repo).
tf broadcaster

Add the following to the CMakeLists.txt file:

```cpp
cmake_minimum_required(VERSION 3.3)
add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)
target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})
```

Type (from the /catkin_ws):

```
catkin_make
```

Copy the start_demo.launch from the shared repo to your package folder.
tf broadcaster

Start your own turtle broadcaster demo and use the tf_echo tool to check if the turtle pose is actually getting broadcast to tf:

    roslaunch learning_tf start_demo.launch

    rosrue tf tf_echo /world /turtle1
Step 3: Writing a tf listener

Follow the tutorial on Writing a tf listener
Make sure to have: src/turtle_tf_listener.cpp (copy the program from the tutorial).
ROS Launch

roslaunch link

ROS provides a mechanism for simultaneously configuring and launching collections of ROS nodes. This is done with lightweight xml files and the roslaunch program.
ROS provides a mechanism for simultaneously configuring and launching collections of ROS nodes. This is done with lightweight XML files and the `roslaunch` program.

Launch files enable users to:

- Associate a set of parameters and nodes with a single file.
- Hierarchically compose collections of other launch files.
- Automatically re-spawn nodes if they crash.
- Change node names, namespaces, topics, and other resource names without recompiling.
- Easily distribute nodes across multiple machines.