tf2

⇒ tf2 is the second generation of the transform library, which lets the user keep track of multiple coordinate frame over time.

⇒ It maintains the relationship between coordinate frames in a tree structure buffered in time.

⇒ It lets the user transform points, vectors etc between any two coordinate frames at any desired point in time.

---

tf: The Transform Library (Paper)

⇒ The tf library has two standard modules:

- Broadcaster
- Listener

★ Data Structure

⇒ Transform and coordinate frames can be expressed as a graph with the transform as edges and the coordinate frame as nodes.

⇒ The graph can exist with one or more disconnected subgraphs.

↳ The transform can be computed between nodes within the subgraph, but not between disconnected subgraphs.

⇒ Transform are inherently directed.
  ↳ To transverse up an edge the inverse of the transform can be used.

⇒ A tree structure also has the benefit of allowing for dynamic changes to the structure without using extra information except the directed graph edges.
  ↳ When an edge is published to a node referencing a different parent node, the tree will resolve to the new parent without extra information.

⇒ Each update to the edge of the tree is specific to the time at which it was measured.

⇒ Queries against the tree are required to have a specific time at which to make the look up.

⇒ To make this possible, a history of the values of an edge of the graph is stored in a chronologically sorted list.

★ Transform Listening

⇒ The Listeners collects the value into a sorted list and when queried can interpolate between the two nearest values.

⇒ The frequency should be selected for the broadcaster which is high enough that Spherical linear interpolation (SLERP) can approximate the motion of the joint between the two samples.

Tutorials

1) Writing a tf2 broadcaster (C++)

```
static tf2_ros::TransformBroadcaster br;
geometry_msgs::TransformStamped tf_s;
br.sendTransform(tf_s);
```

2) Writing a tf2 listener (C++)

```
tf2_ros::Buffer tfBuffer;
tf2_ros::TransformListener tfListener(tfBuffer);
geometry_msgs::TransformStamped tf_s;

try{
  tf_s = tfBuffer.lookupTransform("target_frame",
                                  "source_frame",
                                  ros::Time(0));
} catch(tf2::TransformException &ex) {
  ROS_WARN("%s", ex.what());
  // Handle exception
}
```

The time at which we want to transform

3) tf2 and time

```
tf_S = tfBuffer.lookupTransform("target_frame",
                                "source_frame",
                                , ros::Time::now(),
                                ros::Duration(3.0));
```

→ Optional timeout duration

↳ It will block for up to that duration waiting for it to timeout.