TF (transform) in ROS

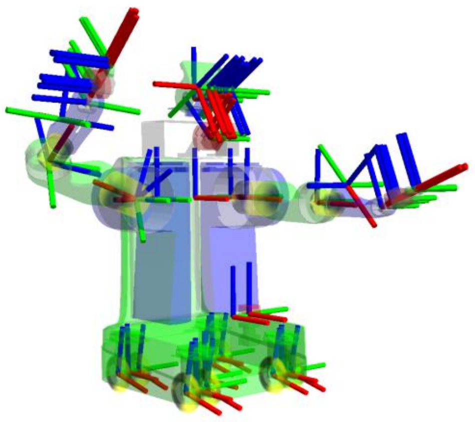
ECET 49900/58100

How Robotics Research Keeps...
Re-Inventing the Wheel
First, someone publishes...
...a paper with a proof-of-concept robot.
This prompts another lab to try to build on this result...
...but they can't get any details on the software used to make it work...
BARF!
...and all the code used by previous lab members is a mess.
So, a grandiose plan is formed to write a new software API...
...and countless sleepless nights are spent writing code from scratch.
But inevitably, time runs out...
...and they write code that barely works but lets them publish...

Credit: PhD comics and Willow Garage

ROS tf (transform) package

- Goal: Maintain relationship between multiple coordinate frames overtime. Transform points, or vectors between two coordinates.



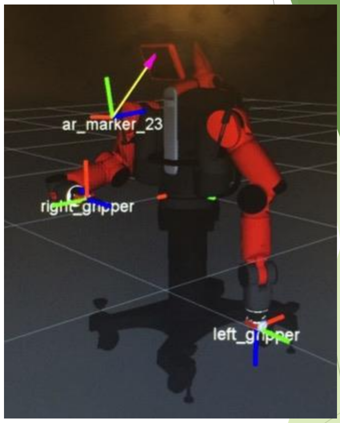
- Published to the system which can be accessed by any node subscribe to it!

- *note tf package is deprecated in favor of the more powerful **tf2_ros** package

Source: http://wiki.ros.org/tf

ROS tf (transform) package

- What are we using it for?
  - Autonomous driving, transform sensor data to map coordinate



- Transform robot coordinates

Source: http://www.kendo.flippen.se/

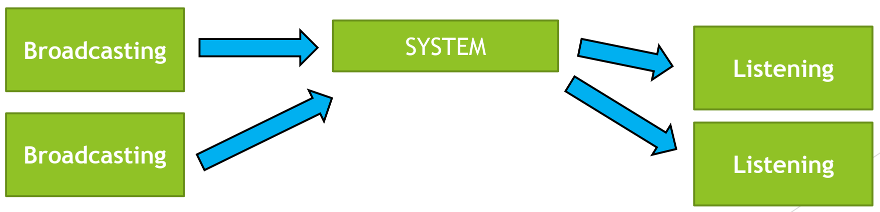
Source: http://web.ics.purdue.edu/~rvoyles/Classes/ROSprogramming/index.html

Using ROS tf (transform) package to transform between coordinates frames

2 main tasks that users generally use tf for transform between coordinates: broadcasting and listening.

Broadcasting transforms:
Publish the relative pose and coordinate to the system
This allow us to setup our own relationship between two coordinate frames

Listening transforms:
Subscribe to published transform and query the specific transform between coordinate frames

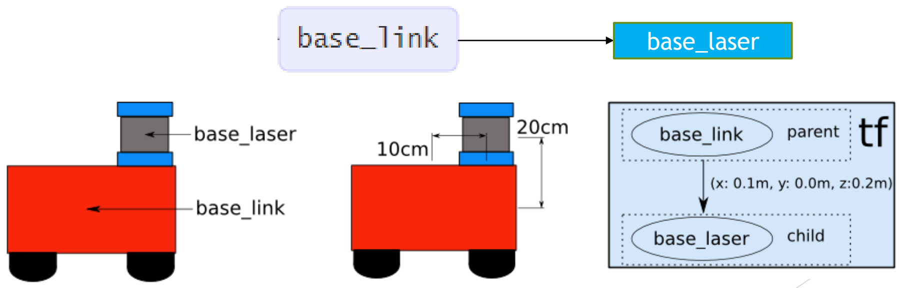


Source: http://wiki.ros.org/tf

Example of using TF broadcasting

Our goal of using the TF broadcasting is to define and establish the relationship between two different coordinate frames, `base_link` and `base_laser`, and build the relationship tree of the coordinate frames in the system.

First step, we have to first define the which is “parent” and “child” because TF defines the “forward transform” as transforming from parent to child. The “inverse transform” goes the other way (and we know how to specify both).



Source: http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF

Example of using TF broadcasting

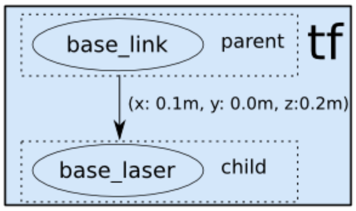
Then, we setup a broadcast:

```
import rospy, tf2_ros, geometry_msgs.msg
def cb(data):
    tfStamp = geometry_msgs.msg.TransformStamped()
    tfStamp.header.stamp = rospy.Time.now()
    tfStamp.header.frame_id = 'base_link'
    tfStamp.child_frame_id = 'base_laser'
    tfStamp.transform.translation = (data.x, data.y, data.z)
    tfStamp.transform.rotation = (data.yaw, data.pitch, data.roll)
    tfBroadcast.sendTransform(tfStamp)

if __name__ == "__main__":
    rospy.init_node("talker")
    rospy.Subscriber('topic_name', 'message_class', cb)
    rospy.spin()
```

name of parent

name of child

Forward transform between them



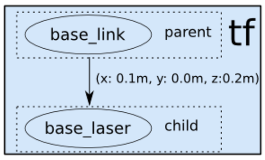
*** See source for full implementation**

Source: http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20broadcaster%20%28Python%29
Source: http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF

Example of using TF listening

TF Listener will access into the existing TF relationship tree and return the relationship between coordinate frames, or even transform points for you.
Then, we setup a listener:

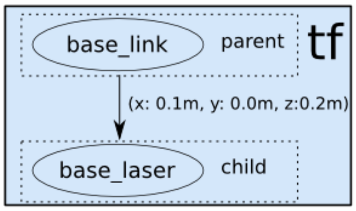
```
import rospy, tf2_ros, geometry_msgs.msg
if __name__ == '__main__':
    rospy.init_node('listen')
    listener = tf.TransformListener()

    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
        try:
            (trans,rot) = listener.lookupTransform('base_link',
            'base_laser', \
                rospy.Time(0))
    # This will give you the coordinate of the child in the parent frame
        except (tf.LookupException, tf.ConnectivityException,
                tf.ExtrapolationException):
            pass
        rate.sleep()
```



*** See source for full implementation**

Source: http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20listener%20%28Python%29
Source: http://mirror.umd.edu/roswiki/doc/diamondback/api/tf/html/python/tf_python.html

Example of using TF listening



Then, we setup a listener:

```
import rospy, tf2_ros, geometry_msgs.msg
if __name__ == '__main__':
    rospy.init_node('listen')
    listener = tf.TransformListener()

    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
        pointstamp = PointStamped()
        pointstamp.header.frame_id = 'base_laser'
        pointstamp.header.stamp = rospy.Time(0)
        pointstamp.point.x = 1.0
        pointstamp.point.y = 2.0
        pointstamp.point.z = 3.0
        try:
            listener.transformPoint('base_link', pointstamp)
        except (tf.LookupException, tf.ConnectivityException,
                tf.ExtrapolationException):
            pass
        rate.sleep()

# This will give you what is the coordinate in parent coordinate frame for (1,2,3) in child.

* See source for full implementation
```

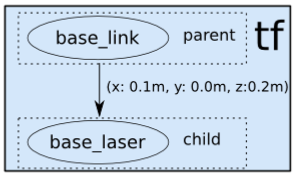
Defining the reference frame…

…of this point

Report it in this frame

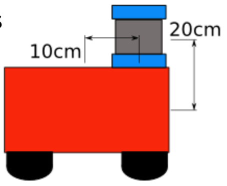
Source: http://wiki.ros.org/tf/Tutorials/Writing%20a%20tf%20listener%20%28Python%29
Source: http://mirror.umd.edu/roswiki/doc/diamondback/api/tf/html/python/tf_python.html

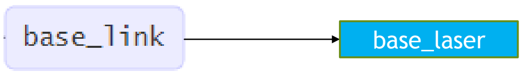
Example of using TF broadcasting staticTF



Another way to broadcast if the transformation is static?
Use "static_transform_publisher" in launch file

static_transform_publisher x y z yaw pitch roll frame_id child_frame_id period_in_ms





Example:

```
<node pkg="tf" type="static_transform_publisher"
name="base2laser" args= "0.1 0.0 0.2 0.0 0.0 0.0
/parent /child 100" />
```

This will transform the parent to the child according to the coordinate transformation input, and publish every 100 ms.

Source: http://wiki.ros.org/tf#static_transform_publisher

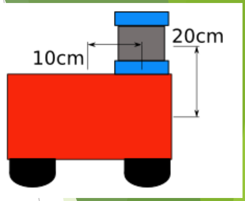
Coordinate frames for Laser Reading

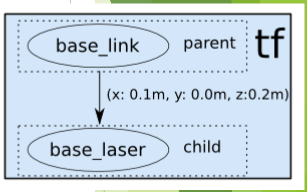
- From ROS Enhancement Proposal (REP) #105
  - Base_link: **Rigidly attached to the mobile robot base.**
  - Odom: **World-fixed coordinate frame.**
    - The pose is continuous (no sudden jump)
    - Accurate in short term local reference! But accumulate errors in long term
  - Map: **World-fixed coordinate frame.**
    - Obtained from re-compute the position from sensor information
    - Not continuous (sudden jump can occurs)

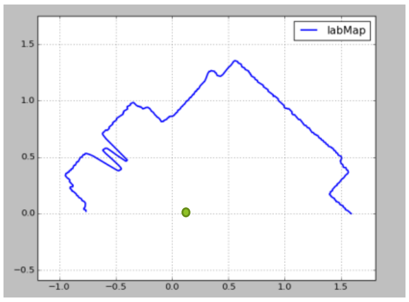
$$\begin{tikzcd}
\text{earth} \arrow[r] & \text{map} \arrow[r] & \text{odom} \arrow[r] & \text{base\_link} \arrow[r] & \text{base\_laser}
\end{tikzcd}$$

Source: http://www.ros.org/reps/rep-0105.html

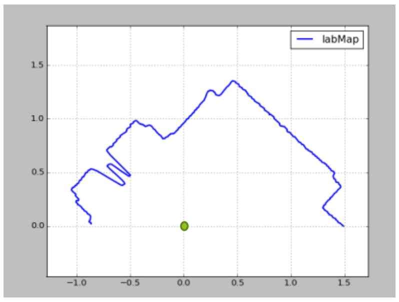
Coordinate frames for Laser Reading







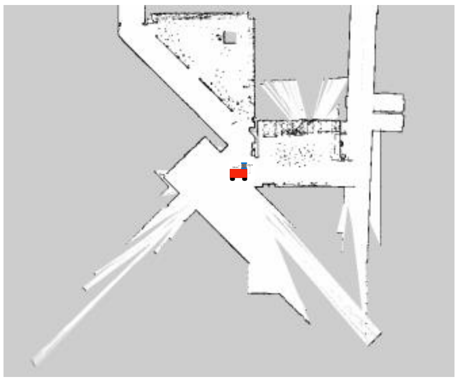
LIDAR reading in base_link coordinate frame



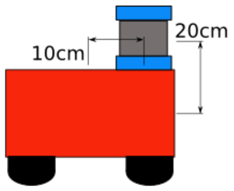
LIDAR reading in base_laser coordinate frame

$$\begin{tikzcd}
\text{base\_link} \arrow[r] & \text{base\_laser}
\end{tikzcd}$$

Coordinate frames for Laser Reading

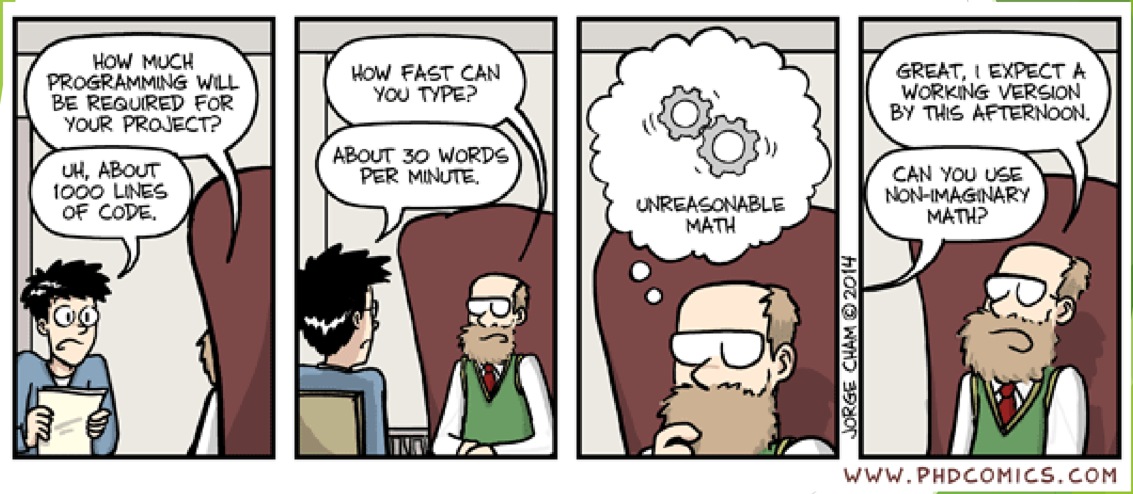


Adjust the TF broadcasting to subscribe to the position/pose of base_link in map coordinate



$$\begin{tikzcd} \text{map} \arrow[r] & \text{base\_link} \arrow[r] & \text{base\_laser} \end{tikzcd}$$

Now, we can implement tf!


HOW MUCH PROGRAMMING WILL BE REQUIRED FOR YOUR PROJECT?
UH, ABOUT 1000 LINES OF CODE.
HOW FAST CAN YOU TYPE?
ABOUT 30 WORDS PER MINUTE.
UNREASONABLE MATH
JORGE CHAM © 2014
GREAT, I EXPECT A WORKING VERSION BY THIS AFTERNOON.
CAN YOU USE NON-IMAGINARY MATH?
WWW.PHDCOMICS.COM

Credit: PhD comics