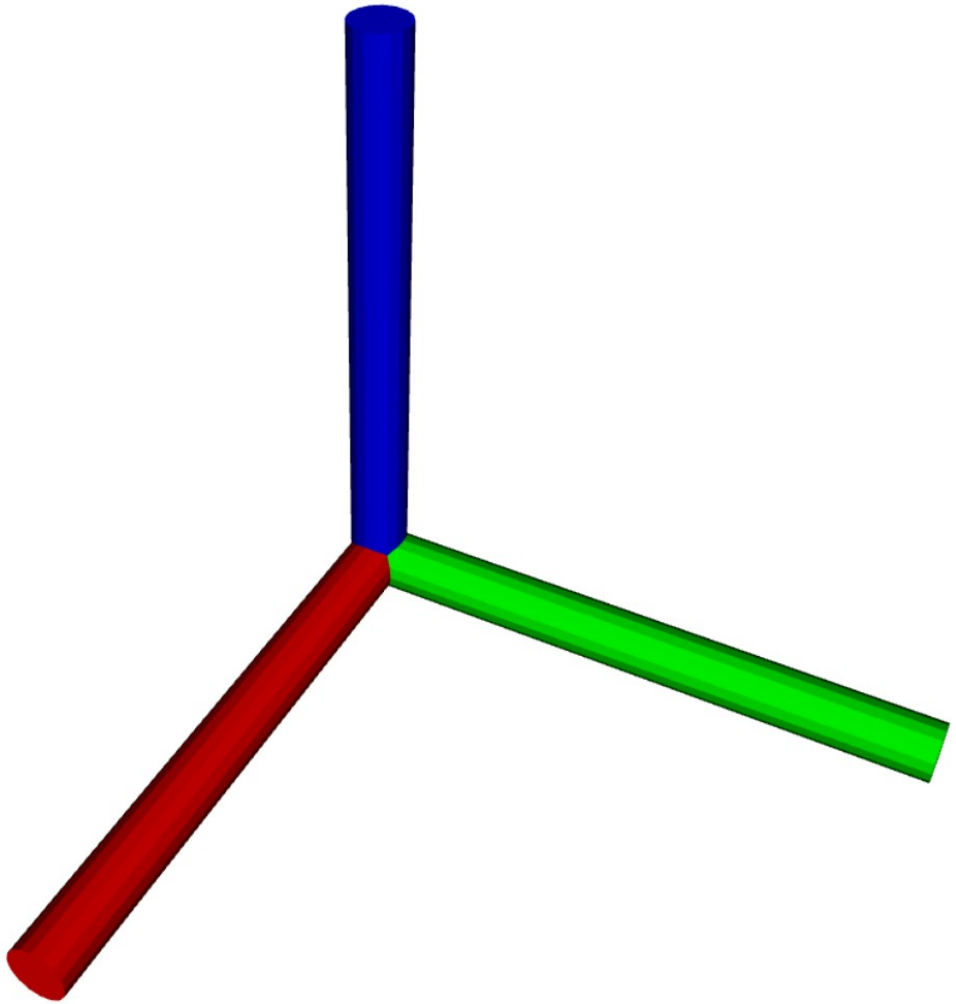
Coordinate Systems

Niklas Fiedler

Coordinate System



Position

Position

$$(x, y, z)$$

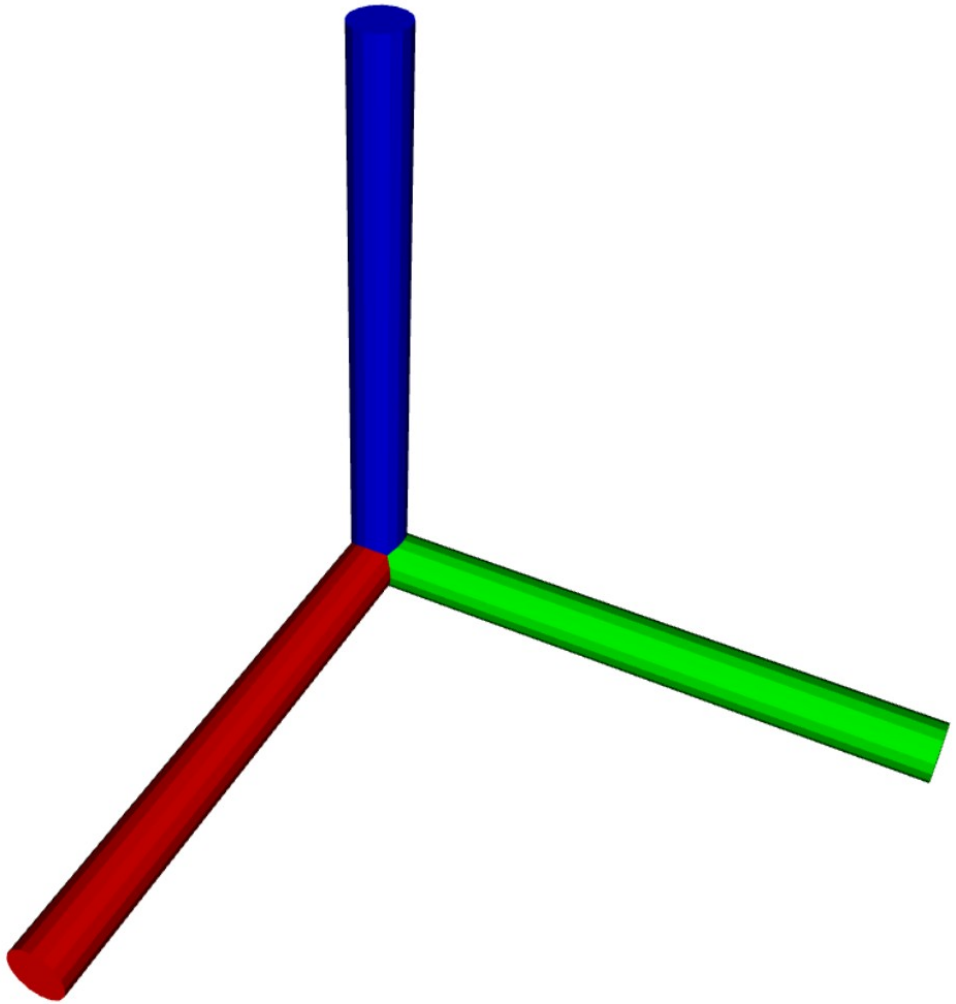
Pose

(Position, Orientation)

Pose

((x, y, z), (x, y, z, w))

Coordinate System



Message Header

std_msgs/Header Message

File: std_msgs/Header.msg

Raw Message Definition

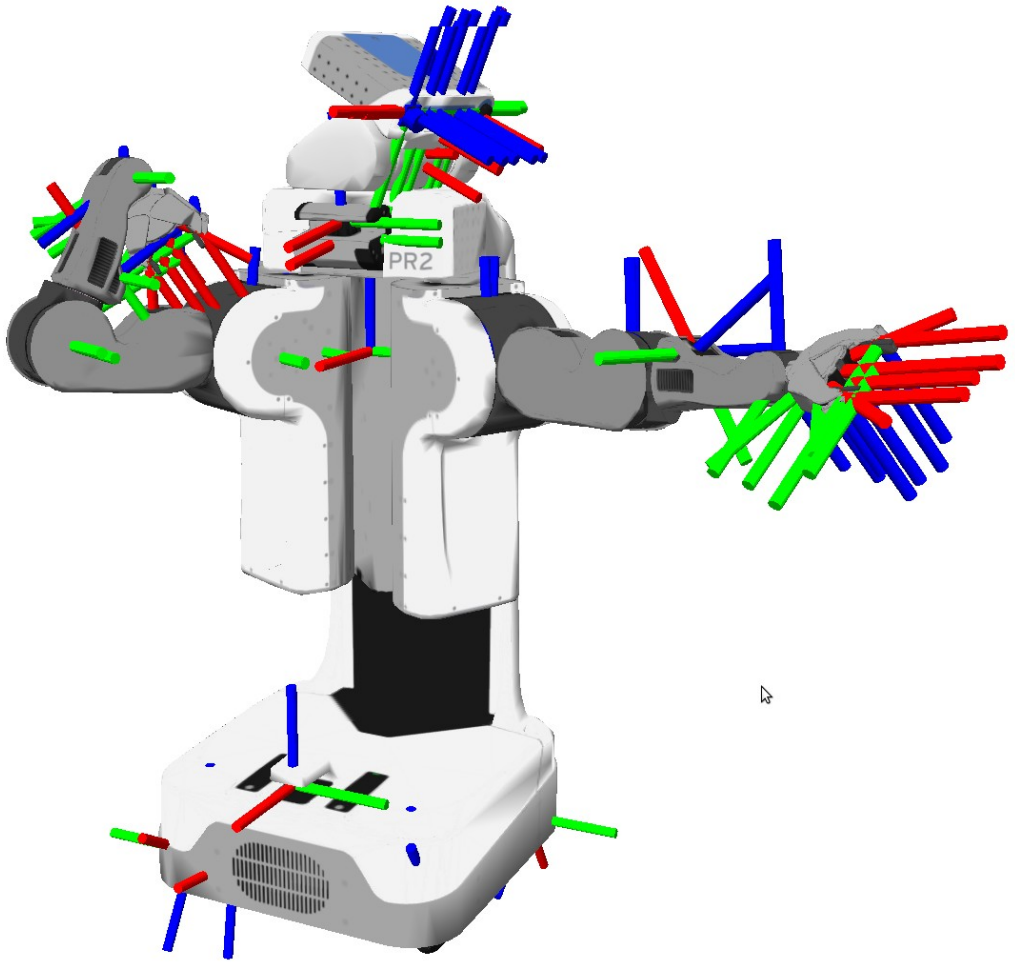
```
# Standard metadata for higher-level stamped data types.
# This is generally used to communicate timestamped data
# in a particular coordinate frame.
#
# sequence ID: consecutively increasing ID
uint32 seq
#Two-integer timestamp that is expressed as:
# * stamp.sec: seconds (stamp_secs) since epoch (in Python the variable is called 'secs')
# * stamp.nsec: nanoseconds since stamp_secs (in Python the variable is called 'nsecs')
# time-handling sugar is provided by the client library
time stamp
#Frame this data is associated with
string frame_id
```

Compact Message Definition

```
uint32 seq
time stamp
string frame_id
```

autogenerated on Wed, 28 Oct 2020 03:35:56

Coordinate Systems



TF 2

- **Successor of TF**
- **Library to handle Transforms**
  - Publish
  - Subscribe
  - Buffer
  - Transform

TF 2

```
#!/usr/bin/env python
import rospy
from apriltag_ros.msg import AprilTagDetectionArray
from geometry_msgs.msg import Twist
import actionlib
import move_base_msgs.msg
import tf2_ros as tf2
from tf2_geometry_msgs import PoseStamped
from math import sqrt
from sensor_msgs.msg import LaserScan, CameraInfo, CompressedImage
import math
from rpc_game_client.srv import PlayerScore
class LaserTag:
    def __init__(self):

        rospy.init_node('laser_tag')

        rospy.sleep(1)
        self.apriltag = None

        self.rpc_game_service = rospy.ServiceProxy("/rpc_score", PlayerScore)
        self.pub = rospy.Publisher('/cmd_vel_mux/input/teleop', Twist, queue_size=10)
        self.twist = Twist()

        self.sub = rospy.Subscriber("tag_detections", AprilTagDetectionArray, self.apriltag_cb)
        self.sub_camera_info = rospy.Subscriber("/camera/rgb/camera_info", CameraInfo, self.camera_info_cb)
        self.sub_compressed_image = rospy.Subscriber("/camera/rgb/image_color/compressed", CompressedImage, self.compressed_image_cb)
        self.compressed_image = None
        self.camera_info = None

        self.tf_buffer = tf2.Buffer(cache_time=rospy.Duration(10))
        self.tf_listener = tf2.TransformListener(self.tf_buffer)
```

TF 2

```
def get_pose(self):
    tag = PoseStamped()
    tag.header = self.apriltag.header
    tag.pose = self.apriltag.detections[0].pose.pose.pose
    tag.header.stamp = rospy.Time(0)

    try:
        tag_bfp = self.tf_buffer.transform(tag, "map", timeout=rospy.Duration(1))
    except (tf2.ExtrapolationException) as e:
        rospy.logwarn(e)
        rospy.logerr('Severe transformation problem concerning the tag!')
        return None
    return tag_bfp
```