Odometry

⇒ It is use of data from motion sensors to estimate change in position over time.

{ Position relative to a starting location }

Navigation

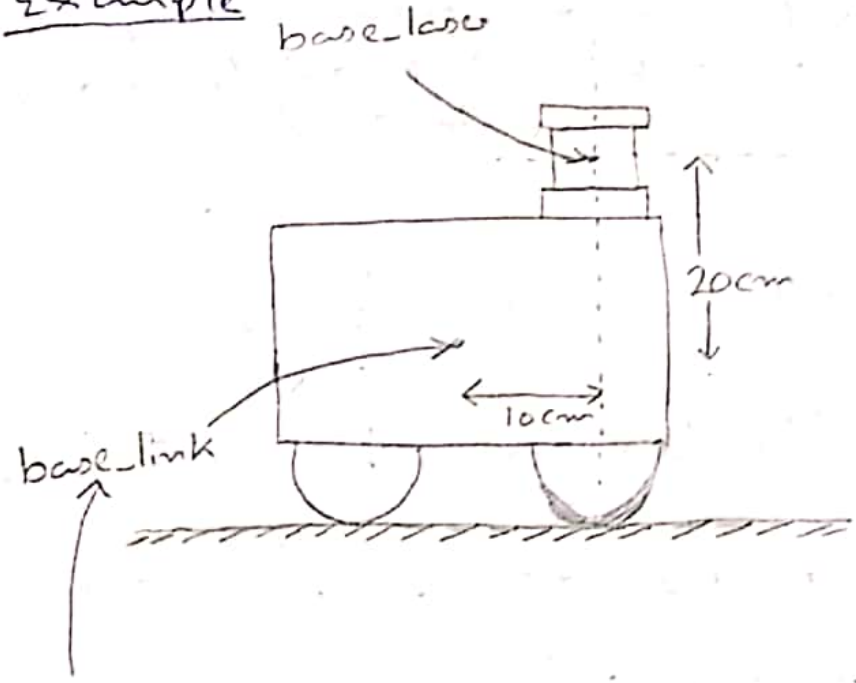
1. Setting up your robot using tf

★ Transform Configuration

⇒ Many ROS packages require the transform tree of a robot to be published using the tf software library.

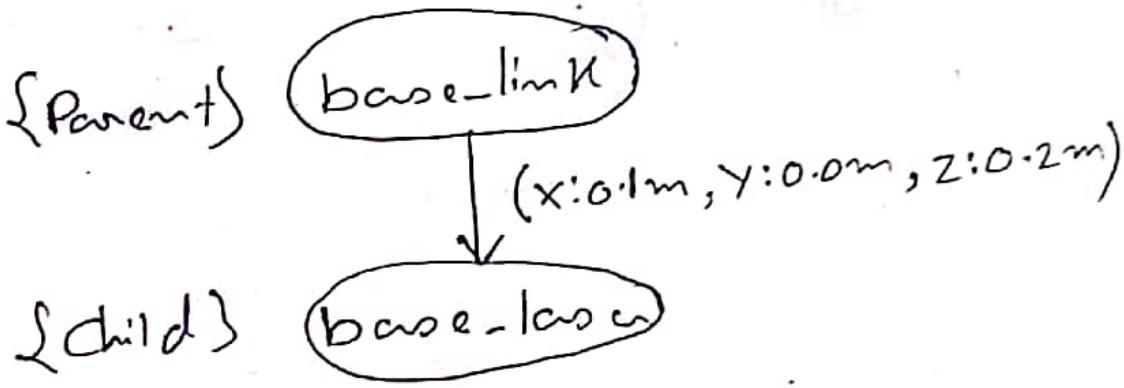
→ { defines offsets in terms of both translation and rotation between different coordinate frames }

Example



For navigation its important that this be placed at the rotational center of the robot

⇒ Each node in the transform tree corresponds to a coordinate frame and each edge corresponds to the transform that needs to be applied to move from the current node to its child.



⇒ Parent and child distinction is important because tf assumes that all transform moves from parent to child.

* Writing Code

* Broadcasting a Transform

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

tf::TransformBroadcaster broadcaster;

while (n.ok()) {
  broadcaster.sendTransform(
    tf::StampedTransform(
      tf::Transform(tf::Quaternion
      (0,0,0,1), tf::Vector3(0.1,0.0,0.2)),
      ros::Time::now(), "base_link",
      "base_laser"));
  r.sleep();
}
```

{Child node}

{Parent node}

* Using a Transform

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

⇒ It includes files that we need to create a tf::TransformListener.

↳ TransformListener object automatically subscribe to the transform message topic over ROS and manages all transform data comming in over the wire.

```
void transformPoint (const tf::Transform
                     Listner & listner)
```

⇒ We'll create a function that take a point in the "base_laser" frame and transforms it to the "base_link" frame.

```
listner.transformPoint ("base_link"
                        , laser_point, base_point);
```

⇒ Give the data in laser_point to base_point with respect to "base_link" frame.

★ Building the Code

CMakeLists.txt

add_executable (tf_broadcaster
src/tf_broadcaster.cpp)

add_executable (tf_listener src/tf_listener.cpp) ★

target_link_libraries (tf_broadcaster ★
${catkin_LIBRARIES})

target_link_libraries (tf_listener
${catkin_LIBRARIES})

2.

Basic Navigation Tuning Guide

★ Range Sensors

↳ Ensure Laser scan data is correct and is comming at expected rate.

★ Odometry

{AMCL}

★ Localization

⇒ First run either gmapping or Karto and joystick the robot around to generate a map.

⇒ Then use this map with AMCL and make sure that the robot stays localized.

* The Costmap
* Local & Global Planner

3.

Setup and Configuration of the Navigation stack on a robot

⇒ The navigation stack requires that the robot be publishing information about the relationships between Coordinate frames using tf.

⇒ The navigation stack assumes that these sensors are publishing either Sensor_msgs/LaserScan or Sensor_msgs/PointCloud messages over ROS.

⇒ The navigation stack requires that odometry information be published using tf and the nav_msgs/Odometry message.

⇒ The navigation stack assumes that it can send velocity commands using a geometry_msgs/Twist message assumed to be in the base coordinate frame of the robot on the "cmd_vel" topic.

⇒ map_server is optional for navigation stack.

4.

Publishing Odometry Information over ROS

⇒ The navigation stack uses tf to determine the robot's location in the world and relate sensor data to a static map.

↳ Tf does not provide any information about the velocity of the robot.

⇒ Because of this, the navigation stack requires that any odometry source publish both a transform and a nav_msgs/Odometry message over ROS that contains velocity information.

2. The nav_msgs/Odometry Message

=> The nav_msgs/Odometry message stores an estimate of the position and velocity of a robot in free space.

Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist