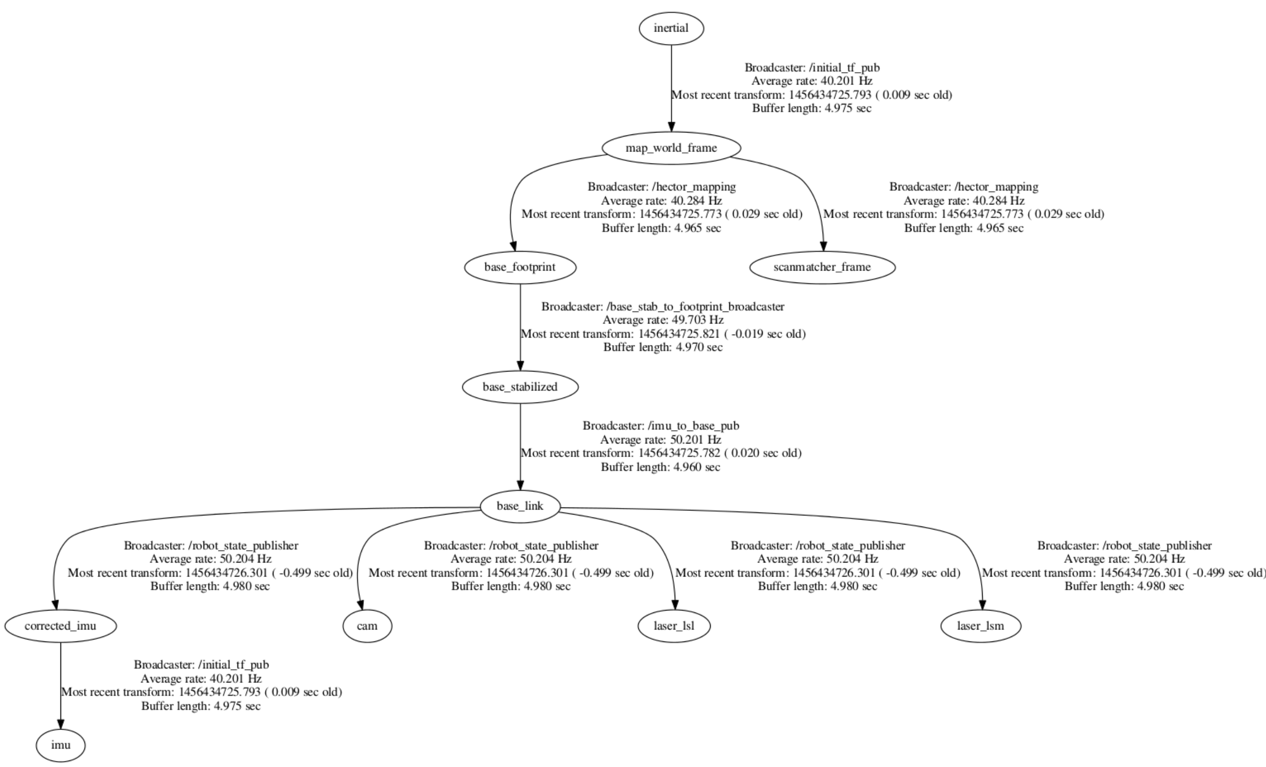
Laser Scanner Tranforms

Brendan Emery

February 2016

Abstract

In this document I outline the relative transforms used on the laser scanning box and how they're calculated/implemented. The final goal is to find the roll and pitch of the of the robot relative to it's stabilised frame, i.e. ${}^{base\_stabilised}R_{\ base\_link}$. This transform is broadcasted and used to transform the incoming laser scans used for localisation and mapping. Refer to `<http://wiki.ros.org/hectoR\_slam/Tutorials/SettingUpForYourRobot>` for a description of the coordinate frame system that I have used. In addition to the standard frames described above, I have also added an imu_corrected frame. This frame will give the imu values corrected by the initial offset of the imu due to any unknown rotations between the IMU and the robot that have not been accounted for in the urdf file. There is also an inertial frame which is determined by the IMU.



1 User Inputs

1.1 Static transform: URDF file

The user must update the urdf file to give the transform ${}^{corrected\_imu}R_{\ base\_link}$. This transform is the rotation between the base_link of the robot and the coordinate frame of the IMU as it sits on the robot. This is done inside "data_recorder/urdf/laser_scanner_description.urdf".

1.2 Static transform: static publisher

The user must use a static_broadcaster_publisher to provide ${}^{I}R_{\ map}$.

The inertial frame will be provided by the IMU's datasheet.

The map frame is provided by the hector mapping package and will be initialised to be aligned and coincident with the base_footprint frame (which is also aligned with the base_link frame at t=0). Therefore the rotation between the inertial and map frame will be given by:

$$ {}^{I}R_{\ map} = {}^{I}R_{\ base\_link},\ when\ t = 0 \tag{1} $$

Therefore,

$$ {}^{I}R_{\ map} = {}^{I}R_{\ base\_link} = {}^{I}R_{\ corrected\_imu} \times {}^{corrected\_imu}R_{\ base\_link},\ when\ t = 0 \tag{2} $$

The user needs to then manually determine these two transforms and then calculate and publish ${}^{I}R_{\ map}$. This is done inside "data_recorder/src/initial_tf_pub.cpp"

2 Find Initial IMU Offset

At time t=0, the user should power up the unit on flat ground. To account for any angular offsets that haven't been accounted for in the transform between the imu and base_link, we use the transform between the imu and the corrected imu frame:

$$ {}^{corrected\_imu}R_{\ imu} = ({}^{I}R_{\ corrected\_imu})^{T} \times {}^{I}R_{\ imu} \tag{3} $$

where,

$$ {}^{I}R_{\ imu} = R_{\ z}(\alpha_0) \times R_{\ y}(\beta_0) \times R_{\ x}(\gamma_0) \tag{4} $$

where γ_0 = roll, β_0 = pitch and α_0 = yaw of the imu in the inertial frame at t=0 and ${}^{I}R_{\ corrected\_imu}$ is calculated above.

We can now apply this transform to the IMU messages to get offset-corrected IMU messages. This is done inside "data_recorder/src/initial_tf_pub.cpp"

3 Find Robot Orientation

We want to find the base_link with respect to the base_stabilised frame at any time to give the roll and pitch values of the robot.

$$
\begin{aligned}
{}^{I}R_{\,base\_link} &= {}^{I}R_{\,corrected\_imu} \times {}^{corrected\_imu}R_{\,base\_link} && (5) \\
&= R_{\,z}(\alpha) \times R_{\,y}(\beta) \times R_{\,x}(\gamma) && (6)
\end{aligned}
$$

where $\gamma =$ roll, $\beta =$ pitch and $\alpha =$ yaw of the robot in the inertial frame.

Since the base_link and base_stabilised frames have the same heading (i.e. their yaw value with respect to the map is the same) and the base_stabilised frame has roll $=$ pitch $=$ 0:

$$
{}^{map}R_{\,base\_stabilised} = R_{\,z}(\alpha) \qquad (7)
$$

So,

$$
\begin{aligned}
{}^{I}R_{\,base\_stabilised} &= {}^{I}R_{\,map} \times {}^{map}R_{\,base\_stabilised} && (8) \\
&= {}^{I}R_{\,map} \times R_{\,z}(\alpha) && (9)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
{}^{base\_stabilised}R_{\,base\_link} &= ({}^{I}R_{\,base\_stabilised})^{T} \times {}^{I}R_{\,base\_link} && (10) \\
&= ({}^{I}R_{\,map} \times R_{z}(\alpha))^{T} \times R_{\,z}(\alpha) \times R_{\,y}(\beta) \times R_{\,x}(\gamma) && (11)
\end{aligned}
$$

This is done inside "data_recorder/src/imu_to_base_pub.cpp"