Transform and visualize SICK Laser data into a different reference frame using MATLAB and tf

Submitted by sprinkle on Sun, 03/12/2017 - 6:04pm

In this tutorial we show how to transform data from a laser sensor into a global reference frame, and then visualize those data using MATLAB's plotting window.

Before you begin

Go ahead and start up the catvehicle_neighborhood.launch file

```
roslaunch catvehicle catvehicle_neighborhood.launch
```

And in a different terminal fire up MATLAB. In (yet another) terminal, be ready to run the hardLeft.sh file

```
cd catvehicle_ws/src/catvehicle/src/tests/openloop
```

Quick overview of TF from a MATLAB tutorial

The tutorial on the MathWorks website here provides a great overview of why to use TF and what it does. To summarize, tf is a tool in ROS that allows you to quickly transform data from one coordinate frame to another, as long as the frames are correctly specified when data are published. With these transforms, we can take advantage of ROS code that figures out the quaternion math to efficiently visualize and analyze data in any coordinate frame---the bonus is that you don't need to know that math!

A MATLAB example to visualize into the odometry frame

The catvehicle/odom frame is used to tell where in the world the catvehicle is located, and what its orientation is. This frame is the equivalent of GPS, but in Gazebo. Data that come from the front laser sensor (front_laser_points) are all in the catvehicle/front_laser_link frame, so as the car moves, these data need to be transformed into the cars reference frame -- or else, it looks like the car is sitting still and everything else is moving around it.

Let's look at the entire file:

```
% tf_scan_tutorial_scan2odom.m
% Author: Jonathan Sprinkle
% Copyright (c) 2017 Arizona Board of Regents
% Based on the tutorial at %
https://www.mathworks.com/help/robotics/ug/transform-laser-scan-data.html
%
% This demonstrates how to transform from a coordinate frame using the tf
% tool that is available in ROS. This particular example shows how this
% works using MATLAB's plotting utility
% NOTE:
% before you start this .m file run in MATLAB:
% rosinit('localhost')
% in a separate tab, make sure you have started up your simulation; if you
% are interested in watching data change in the plots, go ahead and start
% running your controller to make the car move as well.
% get our TF tree
tftree = rostf;
pause(1);
% start up two figures to which we'll be publishing
fig1 = figure;
fig2 = figure;
% Subscribe to /catvehicle/front_laser_points
scansub = rossubscriber('/catvehicle/front_laser_points');
% press ^C to stop the plotting
while 1
% get the latest laser scan
scan = receive(scansub);
% this gives us the rotation/translation with the target frame of odom.
% We want to transform what we see from the front_laser_link into the odom
% frame (which gives it relative to our starting odometry position)
% NOTE: we need to do this every time we visualize, because the car might
% be moving around
tf = getTransform(tftree, '/catvehicle/odom', '/catvehicle/front_laser_link'
% these computations are based on the tutorial
quat = [tf.Transform.Rotation.W,...
        tf.Transform.Rotation.X,...
        tf.Transform.Rotation.Y,...
        tf.Transform.Rotation.Z];
rotm = quat2rotm(quat);
trvec = [tf.Transform.Translation.X,...
         tf.Transform.Translation.Y ...
         tf.Transform.Translation.Z];
tform = trvec2tform(trvec);
tform(1:3,1:3) = rotm(1:3,1:3);
% get cartesian coordinates from the most recent scan
cartScanData = scan.readCartesian;
cartScanData(:,3) = 0;
% convert to homogenous coordiates so we can use rotation/transformation
homScanData = cart2hom(cartScanData);
% put into the target coordinate frame (catvehicle/odom)
trPts = tform*homScanData';
% transform from homogenous back into cartesian frame
```

```
cartScanDataTransformed = hom2cart(trPts');
% now plot the results of the transformations
% into the odometry coordinate frame
figure(fig1)
plot(cartScanDataTransformed(:,1),cartScanDataTransformed(:,2),'ro')
title('odometry coordinate frame');
axis equal
% what the laser sees, with x axies aligned with the car's position
figure(fig2)
title('front\_laser\_link coordinate frame');
plot(cartScanData(:,1),cartScanData(:,2),'b+')
axis equal
% adjust this to be 'faster' if your machine can handle it by setting to,
% say, 0.5 or 0.2
pause(1)
end
```

How to explore from here

Look through the .m file above, to see where the subscription is made to /catvehicle/front_laser_points. That subscription is different than MathWork's tutorial, since our topic is named relative to the laser's position on the car, and it is in the /catvehicle namespace.

Note also that the transform call is using the target frame of catvehicle/odom, which is relative to (0,0) in the Gazebo world. This allows us to project laser data into a common world frame. If you're watching data flash by in this frame, then it should sit still (as long as the environment is not moving).

Visualize the transformed data

In MATLAB try:

```
>> rosinit('localhost');
```

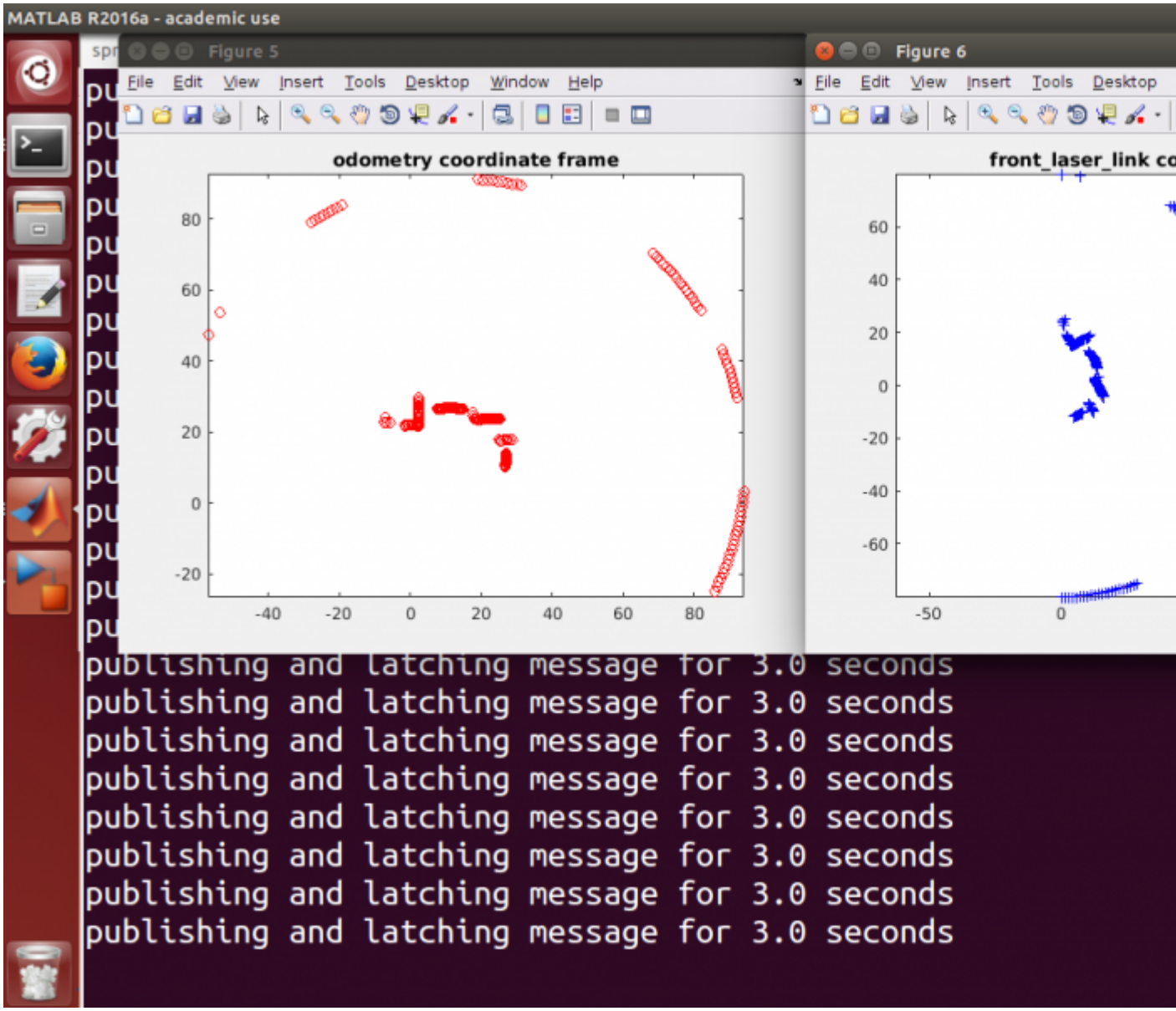
YMMV here, set your IP address or whatever you usually do. Go ahead and (in your tab to run the hardLeft.sh) start up your script to make the car drive around in a circle for awhile:

```
./hardLeft.sh ; ./hardLeft.sh ; ./hardLeft.sh ; ./hardLeft.sh
```

The car should start moving around, so now in MATLAB go ahead and run your .m file

```
>> tf_scan_tutorial_scan2odom
```

You should start seeing two plots flash up around 1Hz. Move them around to see (side by side) something like the below:



The red plot shows a world view where all laser data are projected into the odometry frame of reference. The blue plot shows what the laser is seeing exactly at this point, with the car facing along the x axis and the laser at (0,0) of that frame of reference.

What to try next?

Now you're ready to start interpeting these data and see whether you can (perhaps) infer the kinds of objects they are, based on how big they are or what their shape is.

Backlinks:

- Tutorials

 catvehicle_v2
