tf: On the PR2

ROS + PR2 Training Workshop
What is tf?

A coordinate frame tracking system

- A standardized protocol for publishing transform data to a distributed system
- Helper classes and methods for:
  - Publishing coordinate frame data – TransformPublisher
  - Collecting transform data and using it to manipulate data – Transformer, TransformListener, tf::MessageFilter, ...
tf is Distributed!

- There are two types of tf nodes:
  - Publishers
  - Listeners
- Listeners – listen to /tf and cache all data heard up to cache limit
- Publishers – publish transforms between coordinate frames on /tf
- There is no central source of tf information, or history before a node was started.
Transform Tree

- Each link in the tree is cached
- 10 seconds is default cache time
- It will work with multiple disconnected trees
- Only for transforms within the same tree

view_frames Result
Recorded at time: 1254266629.492

/world

Broadcaster: /turtle2_tf_broadcaster
Average rate: 56.032 Hz
Most recent transform: 0.010 sec old
Buffer length: 4.979 sec

Broadcaster: /turtle1_tf_broadcaster
Average rate: 56.062 Hz
Most recent transform: 0.010 sec old
Buffer length: 4.959 sec

/turtle2
/turtle1
Values of tf

- No data loss when transforming multiple times
- No computational cost of intermediate data transformations between coordinate frames
- The user does not need to worry about which frame their data started
- Information about past locations is also stored and accessible, but not before recording locally was started
Core Methods of Transformer

- **LookupTransform**
  - Get the transform between two coordinate frames

- **CanTransform**
  - Test if a transform is possible between to coordinate frames
How does this work?
1. For both tf data types and message datatypes.
   - TransformPoint
   - TransformVector
   - TransformPose
   - TransformQuaternion

2. Other common message datatypes
   - transformPointcloud
Syncronization Methods

- **WaitForTransform**
  - Block until timeout or transform is available.
- **tf::MessageFilter**
  - Subscribe to a topic and provide the callbacks only when there is enough tf messages to transform the data.
Transformer::waitForTransform

Dangers

No data coming in if in single thread
Blocks all progress, can make all processing late
Protection in Transformer class using SeparateThread method
Using this method can delay the whole system unnecessarily
But it's very convenient for scripting/sequencing
tf::MessageFilter

Purpose

• Provide a non blocking way to queue data pending transform data availability

Usage

• Register a target frame and an incoming topic
• Receive a callback when transforms are available
Object 1 observed in the camera at Time = 0.25 while driving past

Object 2 observed in the camera at Time = 0.5 while driving past

The robot is planning its path at Time = 1 where are all objects?
Object 1 observed in the camera at Time = 0.25 while driving past.

Object 2 observed in the camera at Time = 0.5 while driving past.

The robot is planning its path at Time = 1 where are all objects?
Advanced API

1. Construct transform from first frame to fixed frame at data timestamp
2. Jump to query time in the fixed frame
3. Compute the transform from fixed frame to query frame at query time
4. Return the product of these transforms
Avanced API Walk Through

1) Object Observed

2) Object Transformed into Fixed Frame at Observation Time

3) Object Assumed Static in Fixed Frame

4) Object Transformed into Target Frame at Target Time

5) Object available for manipulation in Target Frame at Target Time
Advanced API examples

1. Compute the position of an observed ball in the target frame at the target time assuming it was stationary in the fixed frame
   - `lookupTransform(ball_frame, ball_time, target_frame, target_time, fixed_frame, result_transform)`

2. Compute how far the robot moved between \( t = 1 \) and \( t = 2 \) in the map frame
   - `lookupTransform(robot_frame, t = 1, robot_frame, t = 2, map_frame, result_transform)`
Debugging Tools

1. Command Line Tools
   - `tf_echo` - Print a specific transform to the screen
   - `tf_monitor` - Display statistics about transforms
   - `roswtf (tf_plugin)` Debug common tf configuration errors
Debugging Tools

1. Visualizations
   - Rviz tf visualization
     - TODO add figure
   - view_frames
tf Python

There are Python bindings for most of the C++ API.

See Docs at: http://www.ros.org/doc/api/tf/html/python

The Python bindings do not have full coverage on c++ methods, in particular the tf::MessageFilter does not have an analog.
Coordinate Frames in the PR2

- Every Link has a `frame_id` the same name as in the URDF
- Each sensor has a `frame_id` in which it takes measurements
- Data is broadcast in the frame in which it was observed.
PR2 Navigation frame_ids

map - The coordinate frame fixed to the map

odom_combined - The self consistent coordinate frame using the odometry measurements only (This will not change on localization updates)

base_footprint - The base of the robot at zero height above the ground

base_link - The base link of the robot
PR2 Sensor Frames

/r(l)_forearm_cam
/wide(narrow)_stereo_r(l)_stereo_camera_frame
/wide(narrow)_stereo_link
/wide(narrow)_stereo_optical_frame
/imu_link
/sensor_mount_link
/high_def_frame
/high_def_optical_frame
/laser_tilt_link
/base_laser_link
PR2 Manipulation Frames

/r(l)_elbow_flex_link       /base_footprint
/r(l)_forearm_cam_frame    /base_laser_link
/r(l)_forearm_cam_optical_frame /base_link
/r(l)_forearm_link         /bl_caster_l_wheel_link
/r(l)_forearm_roll_link    /bl_caster_r_wheel_link
/r(l)_gripper_l_finger_link /bl_caster_rotation_link
/r(l)_gripper_l_finger_tip_link /br_caster_l_wheel_link
/r(l)_gripper_motor_accelerometer_link /br_caster_r_wheel_link
/r(l)_gripper_palm_link    /br_caster_rotation_link
/r(l)_gripper_r_finger_link /double_stereo_link
/r(l)_gripper_r_finger_tip_link /fl_caster_l_wheel_link
/r(l)_gripper_tool_frame   /fl_caster_r_wheel_link
/r(l)_shoulder_lift_link   /fl_caster_rotation_link
/r(l)_shoulder_pan_link    /fr_caster_l_wheel_link
/r(l)_upper_arm_link      /fr_caster_r_wheel_link
/r(l)_upper_arm_roll_link /fr_caster_rotation_link
/r(l)_wrist:flex_link     /head_pan_link
/r(l)_wrist_roll_link     /head_plate_frame
/torso_lift_link          /head_tilt_link
The End

Full documentation at
http://www.ros.org/wiki/tf

Questions?
Tf Challenge

Write a node to draw a visualization_msgs/Marker in rviz on the ground below a detected checkerboard

1. All the other nodes and launch files will be setup.
2. Use the online documentation
3. Setup instructions are on the next page
tf Challenge Setup

1. Install boxturtle unreleased:

   wget --no-check-certificate http://ros.org/roinstall -O ~/roinstall
   chmod 755 ~/roinstall
   . /opt/ros/boxturtle/setup.sh
   ~/roinstall -o ~/boxturtle_wg_devel http://ros.org/rosinstall/l/wg_boxturtle_devel.rosinstall

2. Setup each terminal opened on the robot with:

   . ~/boxturtle_wg_devel/setup.sh

3. Compile and Run with robot started

   • rosmake tf_workshop_demo
   • roslaunch tf_workshop_demo system.launch
tf Challenge Hints

1. There is a topic `/board_pose` provided
2. You will want to subscribe to it with a `tf::MessageFilter`
3. You will then want to transform it into a frame on the ground.
4. You will want to zero out the height.
5. Then create a marker with the resultant pose and broadcast it.
6. Using a C++ class to do this is recommended
7. These slides with the instructions can be found on the workshop wiki page