

Robot Programming with Lisp

7. Coordinate Transformations, TF, ActionLib

Gayane Kazhoyan

Institute for Artificial Intelligence
Universität Bremen

1st December, 2015

Outline

Theory

Coordinate Transformations

TF

ActionLib

Organizational

Theory

Organizational

Outline

Theory

Coordinate Transformations

TF

ActionLib

Organizational

Theory

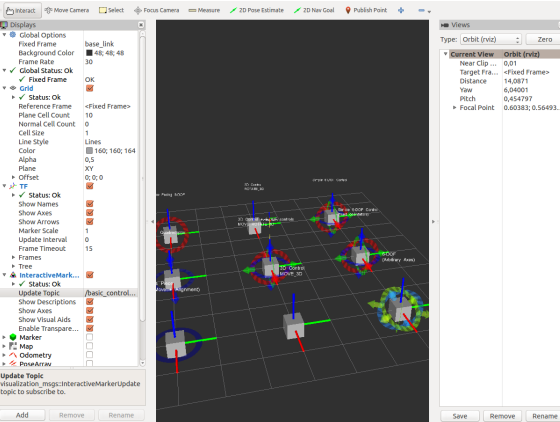
Organizational

Poses in 3D Space

```
$ roscore
```

```
$ rosrn interactive_marker_tutorials basic_controls
```

```
$ rosrn rviz rviz
```



The screenshot shows the RViz interface with the following panels:

- Displays:**
 - Global Options: Fixed Frame (base_link), Background Color (4E, 4E, 48), Frame Rate (30).
 - Global Status: Ok.
 - Grid: Fixed Frame (OK), Status (OK), Reference Frame (<Fixed Frame>), Plane Cell Count (10), Normal Cell Count (0), Cell Size (1), Line Style (Lines), Color (160; 160; 164), Alpha (0.5), Plane (XY), Offset (0; 0; 0).
 - TF: Status (OK), Show Names (checked), Show Axes (checked), Show Arrows (checked), Marker Scale (1), Update Interval (0), Frame Timeout (15), Frames (Tree), InteractiveMarker (checked).
 - InteractiveMarker: Status (OK), Update Topic (/basic_control...), Show Descriptions (checked), Show Axes (checked), Show Visual Aids (checked), Enable Transpare... (checked).
 - Marker (checked), Map (unchecked), Odometry (checked), PoseArrow (checked).
- Views:**
 - Type: Orbit (rviz) - Zero.
 - Current View: Orbit (rviz).
 - Near Clip: 0.01.
 - Target Fra...: <Fixed Frame>.
 - Distance: 14.0871.
 - Yaw: 6.04001.
 - Pitch: 0.454797.
 - Focal Point: 0.60383; 0.56493...
- Time:**
 - ROS Time: 1448961523.94
 - ROS Elapsed: 61489.51
 - Wall Time: 1448961523.97
 - Wall Elapsed: 61489.48
 - Experimental (unchecked)
 - 30 fps

Theory

Organizational

Representing Poses

Point in 3D: $\{x, y, z\}$

3D-Vector

```
CL-TRANSFORMS> (make-3d-vector 1 2 3)
#<3D-VECTOR (1.0d0 2.0d0 3.0d0)>
CL-TRANSFORMS> (describe *)
#<3D-VECTOR (1.0d0 2.0d0 3.0d0)>
 [standard-object]
Slots with :INSTANCE allocation:
 X = 1.0d0
 Y = 2.0d0
 Z = 3.0d0
CL-TRANSFORMS> (y **)
2.0d0
```

Object in 3D: $\{position, orientation\}$

Position: $\{x, y, z\}$

Orientation: axis-angle / rotation matrix / quaternions / ...

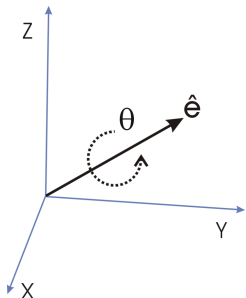
Theory

Organizational

Representing Rotations

Axis-Angle representation:

$$\langle \text{axis}, \text{angle} \rangle = \left\langle \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}, \theta \right\rangle$$



Axis-Angle \rightarrow Quaternion:

$$Q = \begin{pmatrix} q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} = \begin{pmatrix} a_x \sin(\theta/2) \\ a_y \sin(\theta/2) \\ a_z \sin(\theta/2) \\ \cos(\theta/2) \end{pmatrix}$$

3D-Vector

```
CL-TRANSFORMS> (make-quaternion 0 0 0 1)
```

```
CL-TRANSFORMS> (describe *)
```

```
#<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>
[standard-object]
```

```
Slots with :INSTANCE allocation:
```

```
X = 0.0d0
```

```
Y = 0.0d0
```

```
Z = 0.0d0
```

```
W = 1.0d0
```

```
CL-TRANSFORMS> (axis-angle->quaternion
  (make-3d-vector 0 0 1) pi)
```

Theory

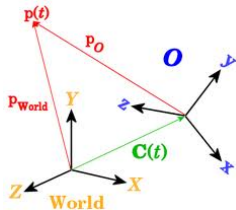
Poses in Lisp

cl-transforms:pose

```
CL-TRANSFORMS> (setf p (make-pose
                        (make-3d-vector 1 2 0)
                        (make-quaternion 0 0 0 1)))

#<POSE
  #<3D-VECTOR (1.0d0 2.0d0 0.0d0)>
  #<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>>
CL-TRANSFORMS> (origin p)
#<3D-VECTOR (1.0d0 2.0d0 0.0d0)>
CL-TRANSFORMS> (orientation p)
#<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>
```

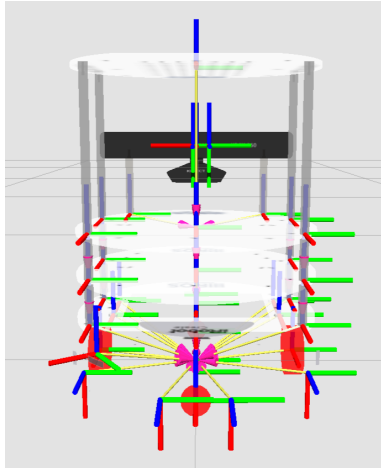
Coordinate Systems



Transformations

```
CL-TRANSFORMS> (setf W (make-identity-pose))
#<POSE
  #<3D-VECTOR (0.0d0 0.0d0 0.0d0)>
  #<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>>
CL-TRANSFORMS> (setf O (make-pose
                        (make-3d-vector 2 0 0)
                        (make-quaternion 0 0 0 1)))
#<POSE
  #<3D-VECTOR (2.0d0 0.0d0 0.0d0)>
  #<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>>
CL-TRANSFORMS> (transform
                (transform-inv (pose->transform O)
                               p))
#<POSE
  #<3D-VECTOR (-1.0d0 2.0d0 0.0d0)>
  #<QUATERNION (0.0d0 0.0d0 0.0d0 1.0d0)>>
```


TurtleBot Coordinate Frames



Theory

Outline

Theory

Coordinate Transformations

TF

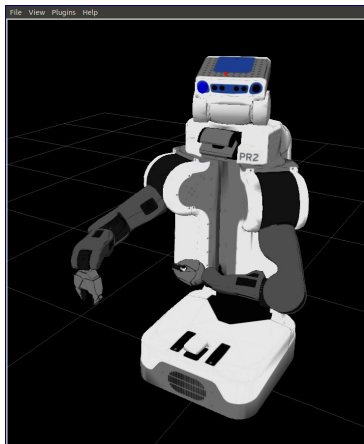
ActionLib

Organizational

Theory

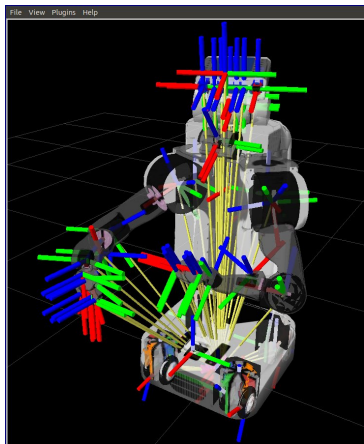
Organizational

Motivation



- Robots consist of many *links*
- Every link describes its own *coordinate system*
- Sensor measurements are local to the corresponding link
- Links change their position over time (including the robot base)

Motivation



- Robots consist of many *links*
- Every link describes its own *coordinate system*
- Sensor measurements are local to the corresponding link
- Links change their position over time (including the robot base)

Implementation

- Transforms are produced by different nodes:
 - Localization in map (AMCL, gmapping)
 - Odometry (base controller)
 - Joint positions (robot controllers and robot_state_publisher)
- Many publishers, many consumers
- Distributed system, redundancy issues, ...

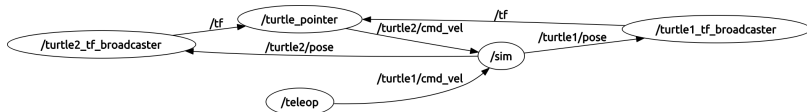


- **TF**: a coordinate frame tracking system
 - Publishing transforms to tf listeners
 - Looking up and calculating transforms by asking tf listeners
- Transformation data is cached over time
- All the transforms together build a TF tree

TurtleSim TF

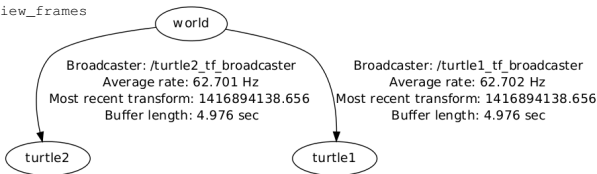
- Launch the turtlesim TF demo:

```
$ roslaunch turtle_tf turtle_tf_demo.launch
```



- Generate a TF tree graph:

```
$ rosrund tf view_frames
```



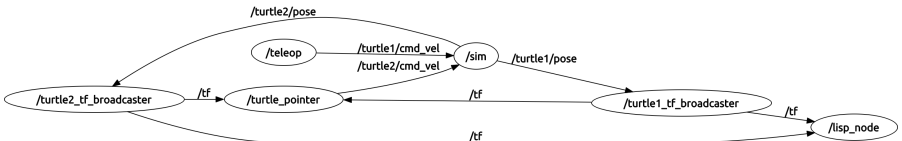
- Listen to transforms:

```
$ rosrund tf tf_echo turtle1 turtle2
```

Lisp TF

cl_tf

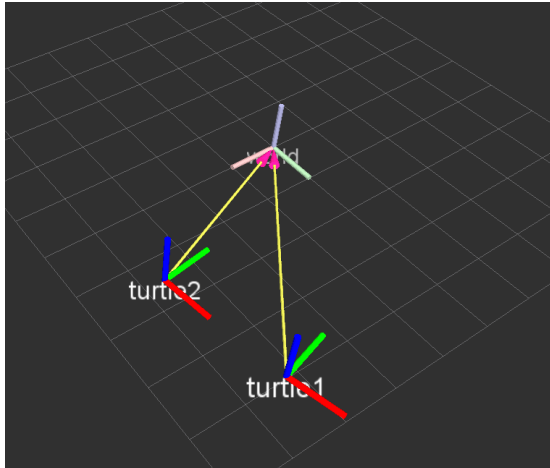
```
TF> (roslisp:start-ros-node "lisp_node")
TF> (defparameter *transform-listener*
      (make-instance 'transform-listener))
TF> (lookup-transform *transform-listener* "turtle1" "turtle2")
#<STAMPED-TTRANSFORM
  FRAME-ID: "turtle1", CHILD-FRAME-ID: "turtle2", STAMP: 1.4169d9
  #<3D-VECTOR (0.0d0 0.0d0 0.0d0)>
  #<QUATERNION (0.0d0 0.0d0 -0.5401331068059835d0 0.8415796022552d0)>>
```



Theory

Organizational

\$ rosrun rviz rviz



Theory

Organizational

Outline

Theory

Coordinate Transformations

TF

ActionLib

Organizational

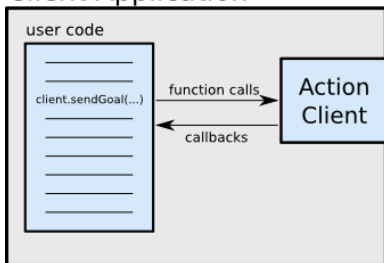
Theory

Organizational

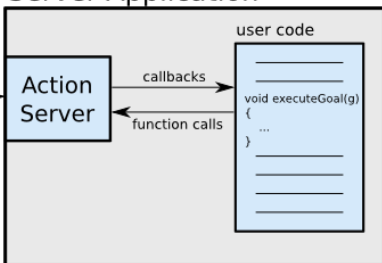
ROS Actions

Interface to define and execute goals:

Client Application



Server Application



ROS

Illustration source: ROS actionlib wiki

Action Protocol

Relies on ROS topics to transport messages.

Action Interface

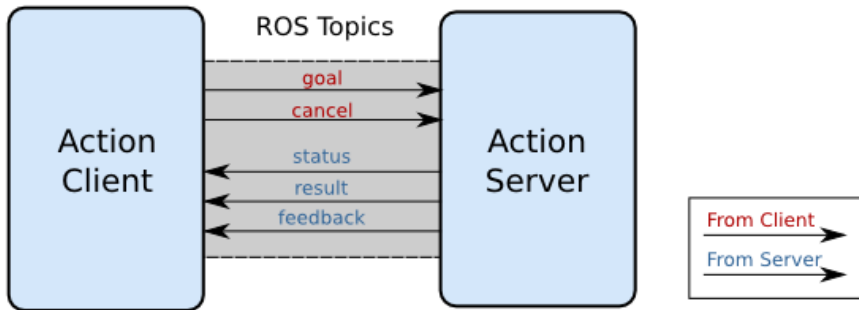


Illustration source: ROS actionlib wiki

Action Definitions

- Similar to messages and services.
- Definition: request + result + feedback
- Defined in *your_package/action/*.action*
- Example: *actionlib_tutorials/Fibonacci.action*

```
# goal definition
int32 order
---
# result definition
int32[] sequence
---
# feedback
int32[] sequence
```

Outline

Theory

Coordinate Transformations

TF

ActionLib

Organizational

Theory

Organizational

Links

- `roslisp_common` repo:
`https://github.com/gaya-/roslisp_common`
- ActionLib Lisp tutorials:
`http://wiki.ros.org/actionlib_lisp/Tutorials`

Info

- Last assignment this week (5 out of 50 points)
- Assignment code: REPO/assignment_7_README.txt
- Next class: 08.12, 14:15, TAB 1.58, bring your laptops!

Q & A

Thanks for your attention!