

# Robot Programming with Lisp

## 8. Coordinate Transformations, TF, ActionLib

Arthur Niedzwiecki

Institute for Artificial Intelligence  
University of Bremen

December 8<sup>th</sup>, 2022

# Outline

Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

**Coordinate Transformations**

TF Library

ActionLib

Organizational

# Outline

Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

**Coordinate Transformations**

TF Library

ActionLib

Organizational

# Intuition

```
$ roscore
```

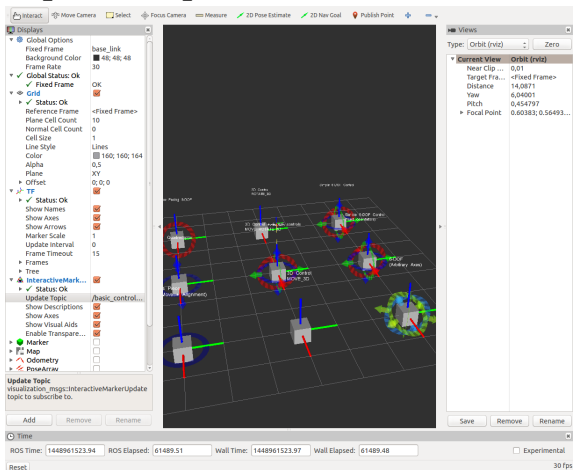
```
$ rosrn interactive_marker_tutorials basic_controls
```

```
$ rosrn rviz rviz
```

```
Fixed Frame: base_link >
```

Add by topic:

InteractiveMarker



The screenshot shows the RViz interface with the InteractiveMarker configuration panel on the left and the 3D visualization of a robot with markers on the right. The configuration panel includes options for Global Options, Grid, TF, and InteractiveMarker. The 3D view shows a robot with several colored markers and arrows indicating its pose and orientation.

Coordinate Transformations

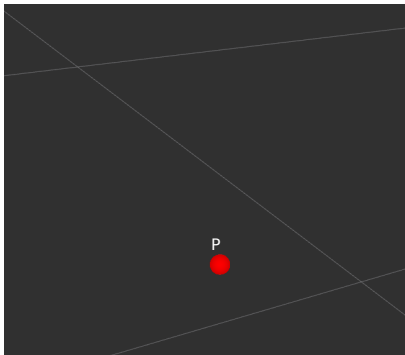
TF Library

ActionLib

Organizational

# 3D Geometry Basics

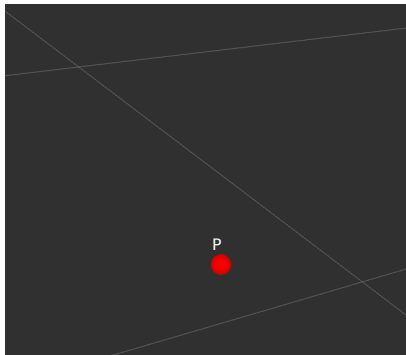
## Coordinates of a point



- What is a point in space? How do we represent it?

# 3D Geometry Basics

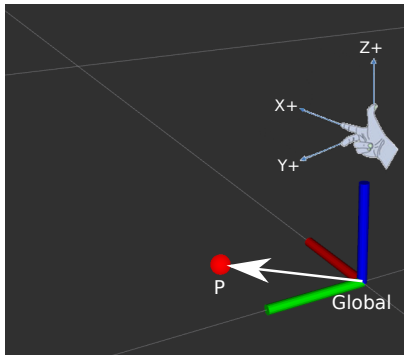
## Coordinates of a point



- What is a point in space? How do we represent it?
- Cartesian coordinates  $(x, y, z)$

# 3D Geometry Basics

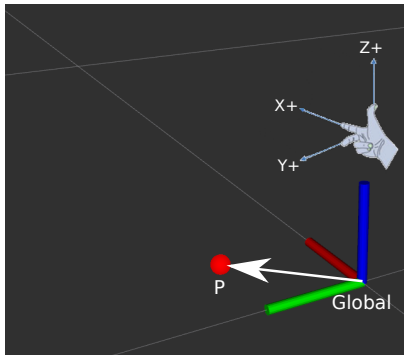
## Coordinates of a point



- What is a point in space? How do we represent it?
- Cartesian coordinates  $(x, y, z)$
- Reference frame  
 $global P = (0.1, 0.1, 0.0)$

# 3D Geometry Basics

## Coordinates of a point



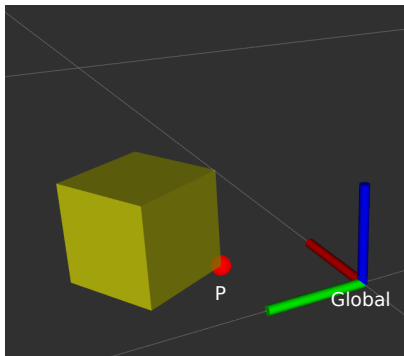
- What is a point in space? How do we represent it?
- Cartesian coordinates  $(x, y, z)$
- Reference frame  
 $_{global}P = (0.1, 0.1, 0.0)$
- Right-hand rule:  
 $(X, Y, Z) \rightarrow (R, G, B)$



# 3D Geometry Basics

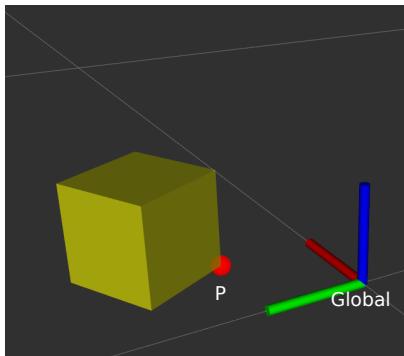
## Coordinates of an object

- How do we represent an object in 3D?



# 3D Geometry Basics

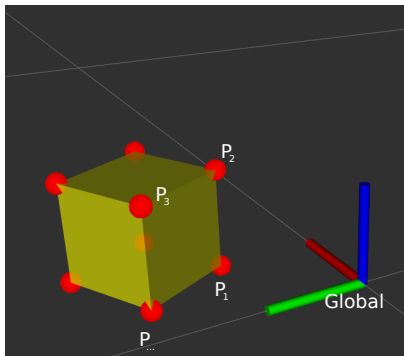
## Coordinates of an object



- How do we represent an object in 3D?
- What is an object?

# 3D Geometry Basics

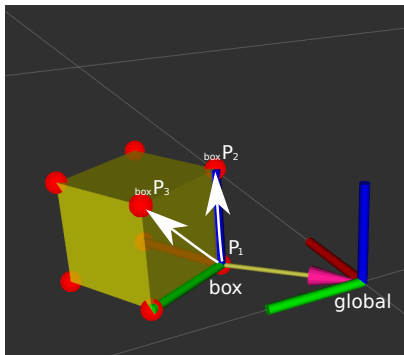
## Coordinates of an object



- How do we represent an object in 3D?
- What is an object?
- Problem: all vertices change coordinates during movement

# 3D Geometry Basics

## Coordinates of an object



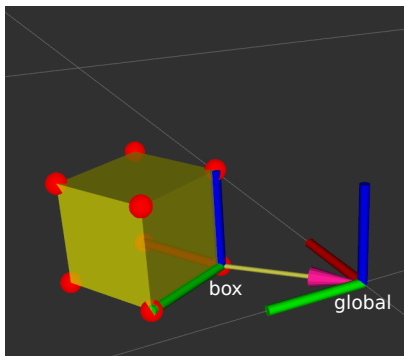
- How do we represent an object in 3D?
- What is an object?
- Problem: all vertices change coordinates during movement
- Solution: describe points on object relative to an object frame

$$global P_1 = (0.1, 0.1, 0.0)$$

$$box P_1 = (0.0, 0.0, 0.0)$$

# 3D Geometry Basics

## Coordinates of an object



- How do we represent an object in 3D?
- What is an object?
- Problem: all vertices change coordinates during movement
- Solution: describe points on object relative to an object frame

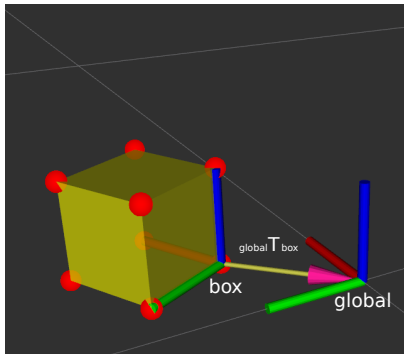
$$global P_1 = (0.1, 0.1, 0.0)$$

$$box P_1 = (0.0, 0.0, 0.0)$$

- What do we need to describe the object frame?

# 3D Geometry Basics

## Coordinates of a frame



- *box* has a position and orientation relative to *global*
- *position & orientation* together are called *pose*
- $global T_{box}$  is a transformation that transforms poses from *box* to *global*
- How do we represent position and orientation?

# Outline

## Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

**Coordinate Transformations**

TF Library

ActionLib

Organizational

# Rotation Representations

There are 4 common ways to describe rotations:

- euler angles
- rotation matrix
- axis-angle
- quaternion



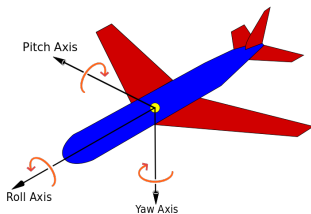
# Euler Angles

- Describes orientation using 3 angles:  
roll (x-rotation), pitch (y-rotation), yaw (z-rotation)

- Rotations are applied in sequence.

What is the sequence is defined through a convention.

There are many conventions, most common are z-y-x, x-y-z and z-x-z



# Euler Angles

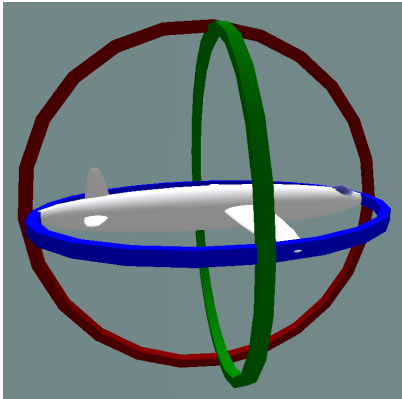
## Pros/Cons

- + easy to interpret
  - has a Gimbal lock problem
  - not suited for interpolation
  - there are many possible conventions, always make sure you know which one is used!
- only useful for user interaction

# Euler Angles

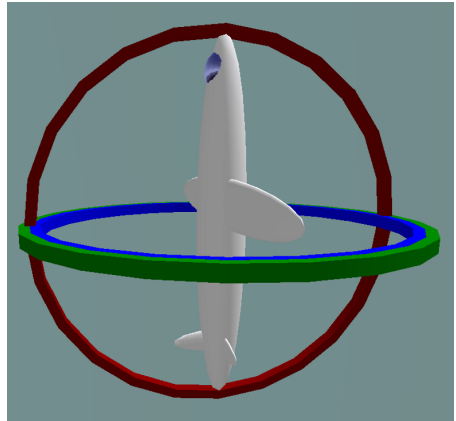
## Gimbal lock

Loss of one degree of freedom, e.g. after  $90^\circ$  pitch (in this case red axis).



Coordinate Transformations

TF Library



ActionLib

Organizational

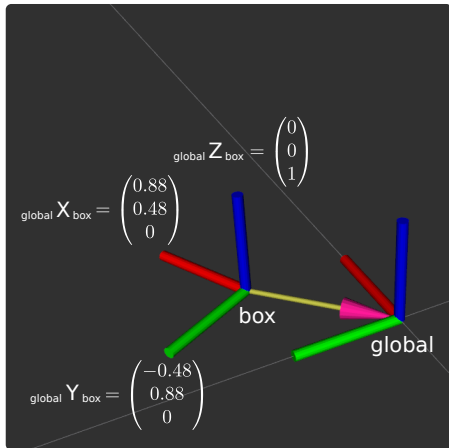
# Rotation Matrix

- $3 \times 3$  matrix  $R$
- is an orthogonal matrix, i.e.  $\det(R) = 1$  and  $R^{-1} = R^T$
- this means, all row (and correspondingly column) vectors are unit vectors, orthogonal to each other

- example:  $R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$  rotates about z-axis by  $\theta$

# Rotation Matrix Interpretation

- example:  $R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$   
rotates about z-axis by  $\theta$
- $global R_{box} = \begin{pmatrix} 0.88 & -0.48 & 0 \\ 0.48 & 0.88 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- columns are axis of box in the global coordinate frame



# Rotation Matrix

## Pros/Cons

- + easiest to do math with
  - rotate a vector with rotation matrix using matrix multiplication
  - rotation matrices can be combined using matrix multiplication
- + easy to construct rotation matrix from 3 vectors
- + can be extended to include translation in 4x4 matrix
  - uses 9 numbers to describe 3 degrees of freedom
  - matrix operations result in buildup of rounding error, you might have to normalize often
  - not suitable for interpolation

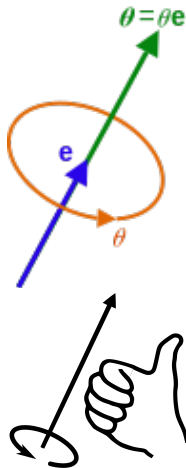
# Axis-Angle

- any rotation can be represented as right hand rotation by  $\theta$  degree about a unit vector  $e$

- angle can be encoded in length of the vector

$$\begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix}, \theta \rightarrow \begin{pmatrix} \theta e_x \\ \theta e_y \\ \theta e_z \end{pmatrix}$$

- can be rotated by rotation matrices using matrix multiplication



# Axis-Angle

## Pros/Cons

- math can get unstable when  $\theta$  is close to 0 or  $\pi$ , because there are infinitely many possible axis
- represents rotation by  $\theta$  differently from  $\theta + 2\pi$ , but it is the same rotation
- + easy interpolation, just scale the angle, but take into account that  $\theta = \theta + 2\pi$
- more useful when describing rotation differences/changes instead of orientations, found in ROS messages like Twist or Accel.



# Quaternion

- $q = (x, y, z, w)$
- number system introduced by Hamilton as an extension of complex numbers, only use case is representation of rotations
- only unit quaternions are used to represent rotations
- can be interpreted as an improved version of axis-angle

- $\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}, \alpha \rightarrow \begin{pmatrix} a_x \cdot \sin(\alpha/2) \\ a_y \cdot \sin(\alpha/2) \\ a_z \cdot \sin(\alpha/2) \\ \cos(\alpha/2) \end{pmatrix}$

# Quaternion

## Pros/Cons

- + in contrast to axis-angle, stable when angle is close to zero and  $\pi$
  - + removes the  $\theta = \theta + 2\pi$  problem from axis-angle
  - + more compact representation than rotation matrices
  - + best for interpolation (slerp algorithm)
  - difficult to interpret
  - most useful for interpolation and describing orientations
- ROS standard for representing poses

# Rotations representations

## Conclusion

- use euler angles only on an interface level
- use axis-angle or quaternion for rigid body dynamics
- use quaternions when storing/sending orientation information or for interpolation
- else use rotation matrices for easy mathematical operations

# Outline

## Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

**Coordinate Transformations**

TF Library

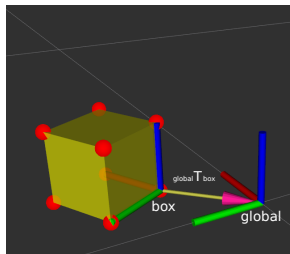
ActionLib

Organizational

# Homogeneous Transformations

- 4 x 4 matrix to represent pose transformations
- ${}_a T_b$  means transform from frame  $b$  to  $a$ , i.e.:  
 ${}_a T_b \cdot {}_b P = {}_a P$
- ${}_a T_b$  is the same as  ${}_a P_b$ , i.e. pose of origin of  $b$  in  $a$
- combined transformation:  
 ${}_c T_b \cdot {}_b T_a = {}_c T_a$
- invertible:  ${}_b T_a^{-1} = {}_a T_b$
- but  ${}_b T_a^{-1} \neq {}_b T_a^T$

$$\begin{array}{c}
 \text{Rotation Matrix} \\
 \left( \begin{array}{ccc|c}
 r_{0,0} & r_{0,1} & r_{0,2} & x \\
 r_{1,0} & r_{1,1} & r_{1,2} & y \\
 r_{2,0} & r_{1,2} & r_{2,2} & z \\
 \hline
 0 & 0 & 0 & 1
 \end{array} \right) \begin{array}{l} \\ \\ \\ \text{Translation} \end{array} \\
 \text{Fixed}
 \end{array}$$

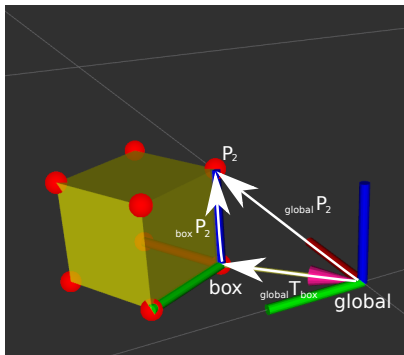


# Homogeneous Transformation

- How do we do  ${}_c T_b \cdot {}_b P = {}_c P$ ?
- Append 1 to point  $P$ , before matrix multiplication:

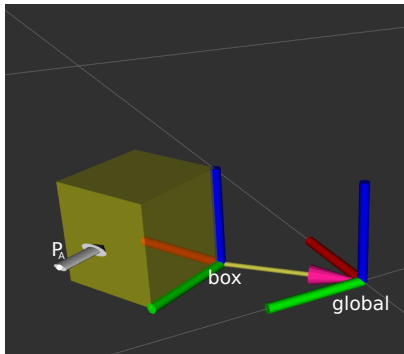
$$\begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} & x \\ r_{1,0} & r_{1,1} & r_{1,2} & y \\ r_{2,0} & r_{2,1} & r_{2,2} & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{0,0}p_x + r_{0,1}p_y + r_{0,2}p_z + x \cdot 1 \\ r_{1,0}p_x + r_{1,1}p_y + r_{1,2}p_z + y \cdot 1 \\ r_{2,0}p_x + r_{2,1}p_y + r_{2,2}p_z + z \cdot 1 \\ 0p_x + 0p_y + 0p_z + 1 \cdot 1 \end{pmatrix}$$

# Homogeneous Transformation



- to transform  ${}_{box}P_2$  into the global frame  ${}_{global}P_2$ , multiply with  ${}_{global}T_{box}$
- ${}_{global}P_2 = {}_{global}T_{box} \cdot {}_{box}P_2$

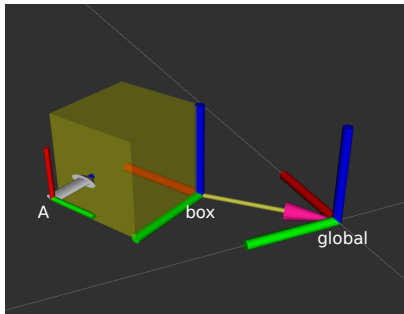
# Homogeneous Transformation



- what is the pose of  $P_A$  in global coordinate frame:  $global P_A$ ?
- choose frame where it is the easiest to express a pose
- $box P_A = (0.05, 0.15, 0.05, 1.0)$
- $global P_A = global T_{box} \cdot box P_A$

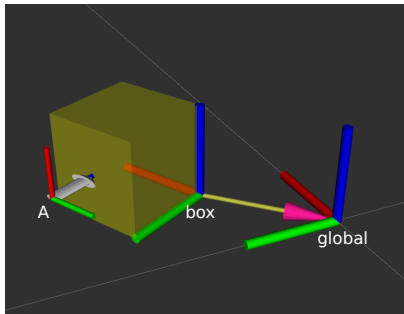


# Homogeneous Transformation



$${}_{box}T_A = \begin{pmatrix} & & & 0.05 \\ & & & 0.15 \\ & & & 0.05 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Homogeneous Transformation



$${}_{box}T_A = \begin{pmatrix} 0 & -1 & 0 & 0.05 \\ 0 & 0 & -1 & 0.15 \\ 1 & 0 & 0 & 0.05 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Points in ROS Lisp

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 / ...

Coordinate Transformations

TF Library

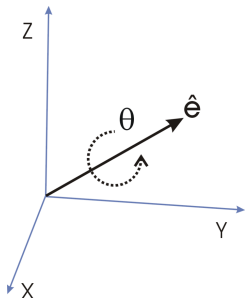
ActionLib

Organizational

# Rotations in ROS Lisp

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)
```

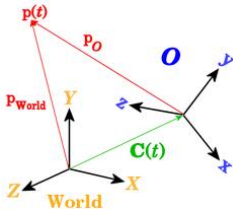
# Poses in ROS Lisp

## 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)>
```

# Transformations in ROS Lisp



## 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)>>
```

# Outline

Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

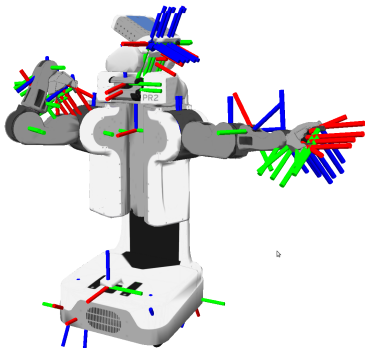
Coordinate Transformations

TF Library

ActionLib

Organizational

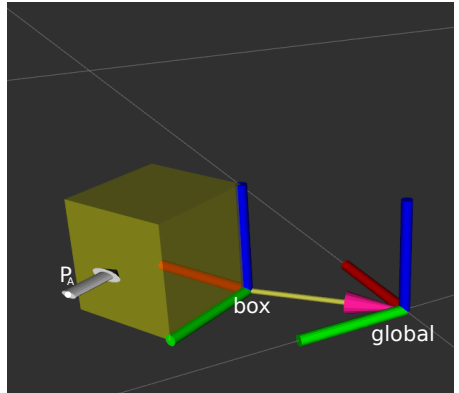
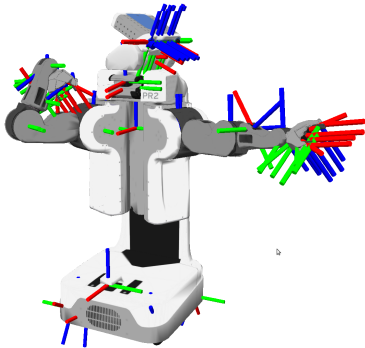
# Motivation



- Robots consist of many *parts* aka *links*
- Each link has its own *coordinate frame*
- Links change their position over time (including the robot base)
- Sensors measurements are defined in their own frame
- Example: transformations from camera to hand coordinates are needed for grasping objects



# Motivation



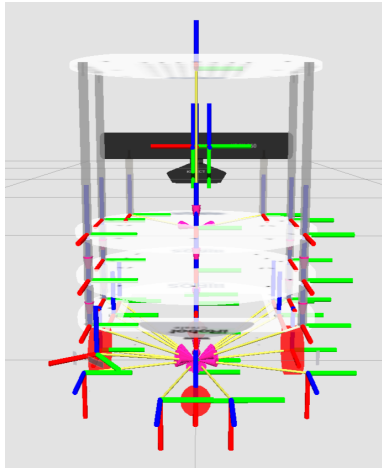
Coordinate Transformations

TF Library

ActionLib

Organizational

# TurtleBot Coordinate Frames



Coordinate Transformations

TF Library

ActionLib

Image courtesy: Yujin Robot  
Organizational

# Tracking Coordinate Frame Changes

- Transforms are produced by different nodes:
  - Localization node (AMCL, gmapping) for finding robot's pose in map
  - Odometry node (base driver) for tracking movement since initial pose
  - Joint positions (robot controllers and robot\_state\_publisher)
- Many publishers, many consumers
- Distributed system, redundancy issues, ...



- **TF**: a coordinate frame tracking system

# What is tf?

transform Library – a distributed coordinate frame tracking system

- Standardized protocol for publishing transforms to tf listeners
- Looking up and calculating transforms by asking tf listeners
- tf listener can be either local Lisp program or global tf buffer
- default global tf buffer is TF2's `buffer_server`
- ROS API for looking up, calculating and sending transforms
- Transforms are published on `/tf` and `/tf_static` topics:

`/tf`

- for all transforms that change over time
- publish with a fixed rate, even if transform didn't change

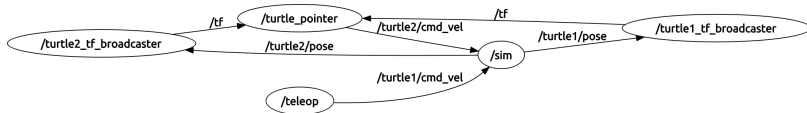
`/tf_static`

- assumed to be static, thus never outdated
- useful for reducing redundancy
- only publish once with latched flag

# TurtleSim TF

Launch the turtlesim TF demo:

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



# Utilities

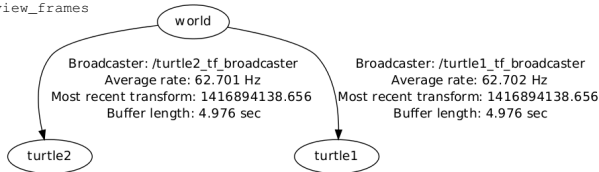
- `view_frames`
- `tf_echo`
- `tf_monitor`
- `static_transform_publisher`
- `RViz`

# Utilities

## roslaunch tf view\_frames

Generate a TF tree graph:

```
$ roslaunch tf view_frames
```



- TF tree consists of frames (links) and the transforms between them.
- Each transform is cached (10 secs default caching time)
- Transforms must form a proper tree (no cycles)
- Can have disconnected trees, but you can only ask for transforms inside of the same tree

# Utilities

## tf\_echo

```
$ rosrn tf tf_echo <source_frame> <target_frame>
```

### tf\_echo

```
$ rosrn tf tf_echo turtle1 turtle2
At time 0.000
- Translation: [0.100, 0.100, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.247, 0.969]
             in RPY (radian) [0.000, -0.000, 0.500]
             in RPY (degree) [0.000, -0.000, 28.648]
```



# Utilities

## `static_transform_publisher`

- `roslaunch tf2_ros static_transform_publisher x y z yaw pitch roll frame_id child_frame_id`  
or  
`roslaunch tf2_ros static_transform_publisher x y z qx qy qz qw frame_id child_frame_id`
- publishes  $global T_{box}$

## `static_transform_publisher`

```
$ roslaunch tf2_ros static_transform_publisher 0.1 0.1 0 3.14 0 0 global box
```

# Utilities

## tf\_monitor

- `roslaunch tf tf_monitor`

### tf\_monitor

```
$ roslaunch tf tf_monitor
RESULTS: for all Frames

Frames:
Frame: turtle1 published by /turtle1_tf_broadcaster Average Delay: 0.000382455 Max Delay: 0...
Frame: turtle2 published by /turtle2_tf_broadcaster Average Delay: 0.000267847 Max Delay: 0...

All Broadcasters:
Node: /turtle1_tf_broadcaster 64.6996 Hz, Average Delay: 0.000382455 Max Delay: 0.000991178
Node: /turtle2_tf_broadcaster 64.7127 Hz, Average Delay: 0.000267847 Max Delay: 0.00133464
```

# TF data types

- `frame_id`: name of the published frame
- `child_frame_id` has to be an existing frame
- `stamp`: time when this transform is valid
- `child_frame_id`  $T_{frame\_id}$

## tf2\_msgs/TFMessage

```
geometry_msgs/TransformStamped[]
transforms
std_msgs/Header header
uint32 seq
time stamp
string frame_id
string child_frame_id
geometry_msgs/Transform transform
geometry_msgs/Vector3 translation
float64 x
float64 y
float64 z
geometry_msgs/Quaternion rotation
float64 x
float64 y
float64 z
float64 w
```

# TF and time

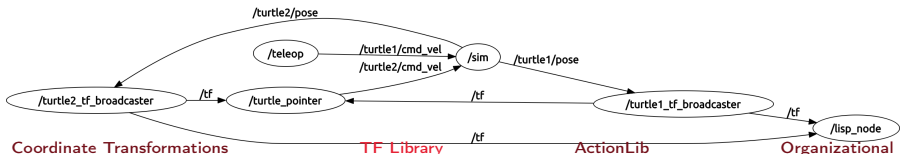
- tf buffers transforms for X seconds
- possible to lookup transforms from the past
- tf interpolates frames
- tf does not extrapolate! it can't see into the future

# Lisp TF

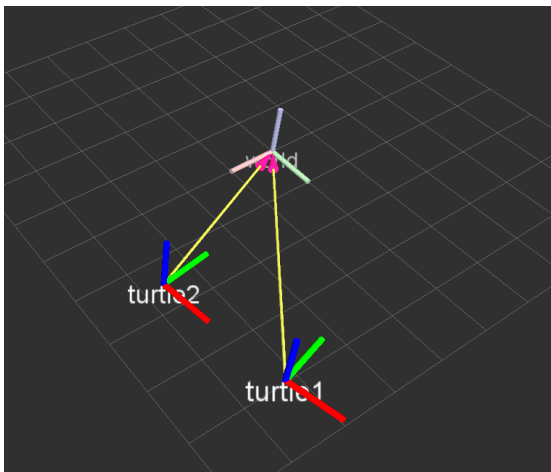
cl\_tf

```
TF> (roslisp:start-ros-node "lisp_node")
TF> (defparameter *transform-listener* nil)
TF> (defun init-listener ()
  (unless (eq (roslisp:node-status) :RUNNING)
    (roslisp:start-ros-node "turtle_follower")
    (setf *transform-listener* (make-instance 'transform-listener))))
TF> (lookup-transform *transform-listener* :source-frame "turtle1"
  :target-frame "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)>>
```



# \$ rosrn rviz rviz



Coordinate Transformations

TF Library

ActionLib

Organizational

# Outline

Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

Coordinate Transformations

TF Library

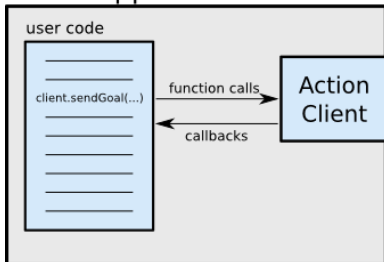
ActionLib

Organizational

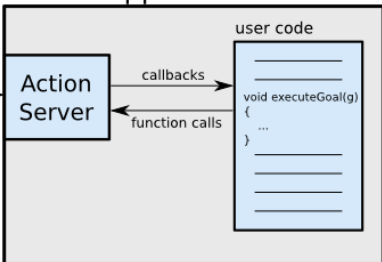
# ROS Actions

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

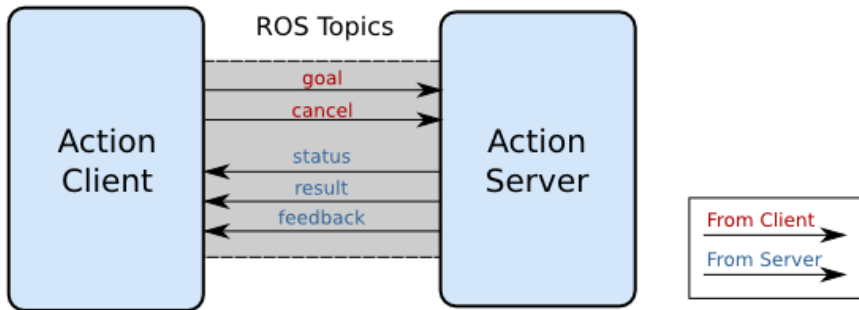


Illustration source: ROS actionlib wiki

# 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
```

# Links

- Gilbert Strang's MIT course on linear algebra (free access):

<https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/>

- Also check out the 3D-Poses tutorial in cram-teaching:

[https://github.com/cram2/cram\\_teaching/blob/main/lectures/tutorials/02-Lesson\\_02\\_CRAM\\_Basics.ipynb](https://github.com/cram2/cram_teaching/blob/main/lectures/tutorials/02-Lesson_02_CRAM_Basics.ipynb)

# Outline

Coordinate Transformations

3D Geometry Basics

Rotation Representations

Homogeneous Transformations

TF Library

ActionLib

Organizational

Coordinate Transformations

TF Library

ActionLib

Organizational

# Assignment 7

## Update

- Update repository:

```
$ git pull
```

- Merge/Edit `docker-compose-headless.yml` to use image version 1.2

- Compare your changes with those on GitHub

```
https://github.com/cram2/cram\_teaching/blob/main/docker-compose-headless.yml
```

- Launch the container

```
$ docker-compose -f docker-compose-headless.yml up
```

- Attach to container

```
$ docker exec -it cram_headless /bin/bash
```

# Assignment 7

## Setup workspace

- Go to assignment 7 workspace

```
$ cd /home/lectures/robot_programming_with_lisp/07_turtle_chase/ros_ws/
```

- Build workspace

```
$ catkin_make
```

- Add source-command at the bottom of /root/.bashrc

```
$ nano /root/.bashrc
```

```
source /home/lectures/robot_programming_with_lisp/07_turtle_chase/ros_ws/devel/setup.bash
```

- Check if the workspace is sourced with

```
$ echo $CMAKE_PREFIX_PATH
```

# Byobu

Start byobu: `$ byobu`

- F2 - new terminal
- CTRL-D - kill current terminal
- F3 - previous terminal
- F4 - next terminal
- F7 - unlock scroll function. Hit ENTER to end.
- F8 - rename current terminal

See also the Byobu Cheat Sheet:

[https://cheatography.com/mikemikk/cheat-sheets/byobu-keybindings/pdf\\_bw/](https://cheatography.com/mikemikk/cheat-sheets/byobu-keybindings/pdf_bw/)

# Showcase turtle chase

Task for assignment 7 looks like this

```
$ roslaunch turtle_tf turtle_tf_demo.launch  
except that turtle1 automatically moves in random shapes
```



# Info

- Assignment points: 7 points
- Due: 14.12.22
- TF Lisp tutorial:  
[http://wiki.ros.org/cl\\_tf/Tutorials/clTfBasicUsage](http://wiki.ros.org/cl_tf/Tutorials/clTfBasicUsage)
- ActionLib Lisp tutorial (Section 1 and 2, not 3):  
[http://wiki.ros.org/actionlib\\_lisp/Tutorials/actionlibBasicUsage](http://wiki.ros.org/actionlib_lisp/Tutorials/actionlibBasicUsage)
- Next class: 15.12, 14:15

# Q & A

Thanks for your attention!