TIAGo Training Sessions

Upper Body Motions

Upper body motions

Introduction

PAL ROBOTICS

Upper body joints
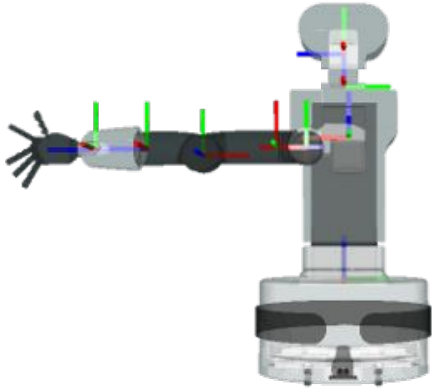
Joint	Type	Lower limit	Upper limit
torso	prismatic	0 mm	350 mm
head_1	revolute	-75º	75º
head_2	revolute	-60º	45º
arm_1	revolute	0	157.5º
arm_2	revolute	-90º	62.5º
arm_3	revolute	-202.5º	90º
arm_4	revolute	-22.5º	135º
arm_5	revolute	-120º	120º
arm_6	revolute	-90º (*)	90º (*)
arm_7	revolute	-120º	120º

(*) Wrist version with force/torque sensor arm_6 range is [-81º, 81º]

PΛL
ROBOTICS

Motion types

Type of motion	Interfaces	Joints supported
Playback of **prerecorded motions**	● Action Server	● torso ● arm ● hand / gripper ● head
Joint trajectory	● Topic ● Action Server	● torso ● arm ● hand / gripper ● head
Position increments	● Action Server	● torso ● head

PAL ROBOTICS

Motions executed by joystick

Motion execution with joystick (I)

Buttons	Description	Motion type	Self-collisions check
LB	Torso upwards	position increment	✓
LT	Torso downwards	position increment	✓

Motion execution with joystick (II)

Buttons	Description	Motion type	Self-collisions check
🔵	Move head leftwards	Position increment	✓
🔴	Move head rightwards	Position increment	✓
🟢	Move head downwards	Position increment	✓
🟠	Move head upwards	Position increment	✓

PAL ROBOTICS

Motion execution with joystick (III)
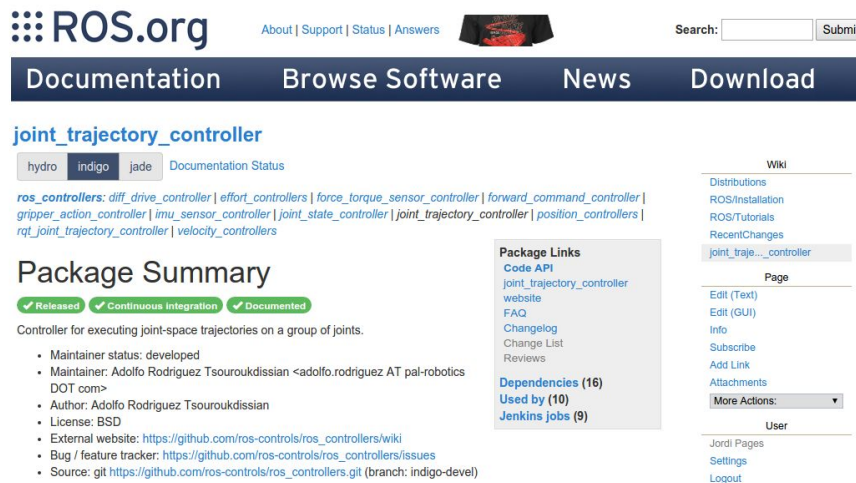
Buttons	Description
RB	Open hand / gripper
RT	Close hand / gripper

PAL ROBOTICS

Joint trajectory motions

PAL ROBOTICS

Joint trajectory motions

- Executing joint-space trajectories on a group of joints

- Underlying sytem: **joint_trajectory_controller** ROS package:

Joint trajectory motions specification (I)

- Trajectories are specified as a set of waypoints to be reached at specific time instants

trajectory_msgs/JointTrajectory:

      Header header
      string[] joint_names
      JointTrajectoryPoint[] points

trajectory_msgs/JointTrajectoryPoint:

      # Each trajectory point specifies either positions[, velocities[, accelerations]]
      # or positions[, effort] for the trajectory to be executed.
      # All specified values are in the same order as the joint names in JointTrajectory.msg

      float64[] positions
      float64[] velocities
      float64[] accelerations
      float64[] effort
      duration time_from_start

PAL ROBOTICS

Joint trajectory motions specification (II)

- Example of trajectory_msgs/JointTrajectory msg:

```
header:
  seq: 83
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
joint_names: ['head_1_joint', 'head_2_joint']
points:
  -
    positions: [-0.10506145631511613, -0.04555309342499999]
    velocities: []
    accelerations: []
    effort: []
    time_from_start:
      secs: 0
      nsecs: 100000000
```

PAL ROBOTICS

Joint trajectory motions interfaces

- **Topic interfaces** (trajectory_msgs/JointTrajectory)

  /torso_controller/command

  /arm_controller/command

  /head_controller/command

  /hand_controller/command

  /gripper_controller/command

  /torso_controller/safe_command ⎫ executes motion only if it does not lead
  /arm_controller/safe_command   ⎭ to a self-collision

- **Action interfaces** (control_msgs::FollowJointTrajectoryAction)

  /torso_controller/follow_joint_trajectory

  /head_controller/follow_joint_trajectory

  /arm_controller/follow_joint_trajectory

  /hand_controller/follow_joint_trajectory

  /gripper_controller/follow_joint_trajectory

  /safe_arm_controller/follow_joint_trajectory   ⎫ executes motion only if it does not lead
  /safe_torso_controller/follow_joint_trajectory ⎭ to a self-collision

trajectory_msgs/JointTrajectory trajectory
JointTolerance[] path_tolerance
JointTolerance[] goal_tolerance
duration goal_time_tolerance

Goal

---
int32 error_code
string error_string

Result

Header header
string[] joint_names
trajectory_msgs/JointTrajectoryPoint desired
trajectory_msgs/JointTrajectoryPoint actual
trajectory_msgs/JointTrajectoryPoint error

Feedback

PAL ROBOTICS

TIAGo T.S.

Joint trajectory motions execution (I)

- **Command line execution** (topic interface)**:**

```
rostopic pub -1 /head_controller/command trajectory_msgs/JointTrajectory
"header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
joint_names: ['head_1_joint','head_2_joint']
points:
- positions: [-0.2, 0.5]
  velocities: []
  accelerations: []
  effort: []
  time_from_start: {secs: 2, nsecs: 0}"
```

PAL ROBOTICS

Joint trajectory motions execution (II)

- **rqt GUI interface** (topic interface):

export ROS_IP=10.68.0.128 ————————————————→ specify the IP of your computer

rosrun rqt_joint_trajectory_controller rqt_joint_trajectory_controller
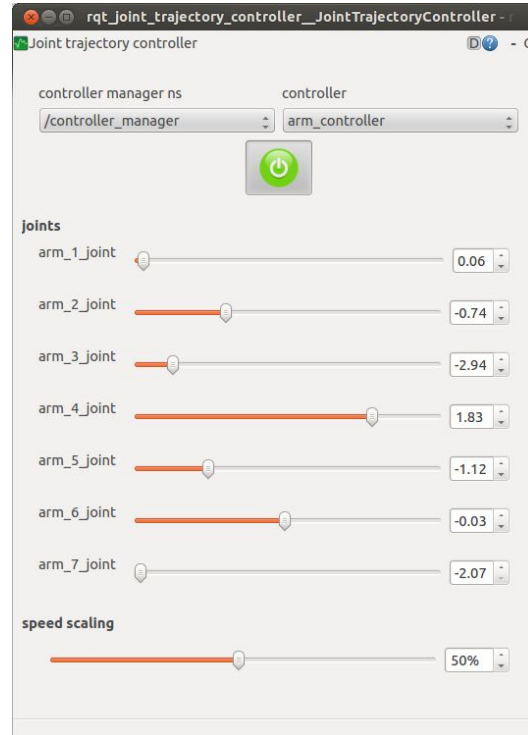
/arm_controller/command

/torso_controller/command

/head_controller/command

/gripper_controller/command

TIAGo T.S.

P∧L
ROBOTICS

Joint trajectory motions execution (III)

- **Web server interface** (topic interface) in the **Control joints** tab

For the **torso** and the **arm** the safe joint trajectory topic interfaces are used:

collision check OK

/torso_controller/safe_command → /torso_controller/command

collision check OK

/arm_controller/safe_command → /arm_controller/command

PAL ROBOTICS

Joint trajectory motions execution (IV)

- **move_joint** (action interface)
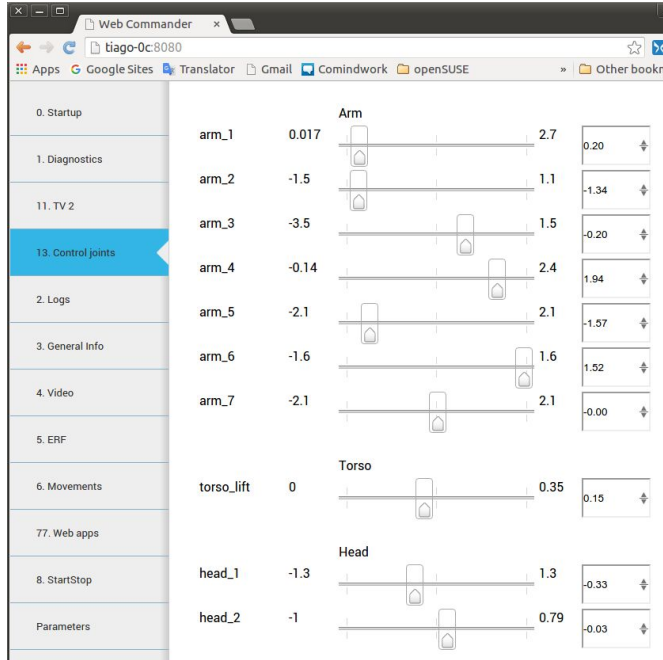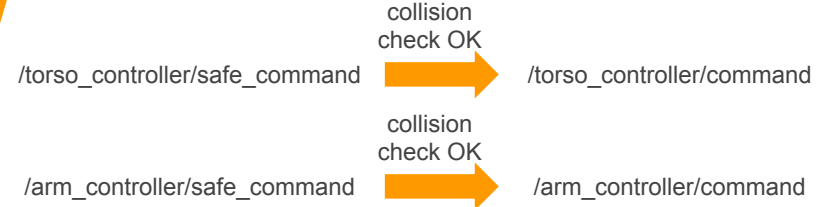
- The node **move_joint** in play_motion package can be used to move any individual joint of the upper body

- Examples:

  rosrun play_motion move_joint torso_lift_joint 0.18 2

  → **move_joint** sends the required motion to the corresponding Action Server: */torso_controller/follow_joint_trajectory*

  rosrun play_motion move_joint head_2_joint 0.2 3

  → **move_joint** sends the required motion to the corresponding Action Server: */head_controller/follow_joint_trajectory*

  WARNING: this node does not check if the new position is in collision

TIAGo T.S.

PAL ROBOTICS

Predefined upper body motions

PAL ROBOTICS

Predefined motions

- Predefined motions involving a subset of these DoF can be played back at any time

- Underlying system for upper body predefined motions: **play_motion** ROS package:

http://wiki.ros.org/play_motion

Predefined motions specification (I)

- **Motion specification** format

  - *joints*: list of joints used in the motion
  - *points*: list of robot states
    - *positions*: list of joint positions w.r.t. joints list
    - *time_from_start*: time given to reach the position
  - meta: meta information that can be used by other applications

- Predefined motions:

  tiago_bringup/config/tiago_motions.yaml

  - home
  - unfold_arm
  - reach_floor
  - reach_max
  - head_tour
  - wave
  - pregrasp_weight
  - do_weights
  - pick_from_floor
  - shake_hands

  - open_hand
  - close_hand
  - pointing_hand
  - gun_hand
  - thumb_up_hand
  - pinch_hand

P∧L
ROBOTICS

TIAGo T.S.

Predefined motions specification (II)

- Motion specification example:

`rosparam get /play_motion/motions/wave -p`

Web commander

http://tiago-0c:8080

6. Movements   Wave

```
joints: [arm_1_joint, arm_2_joint, arm_3_joint, arm_4_joint, arm_5_joint, arm_6_joint, arm_7_joint]
meta:
  description: wave
  name: Wave
  usage: demo
points: - positions: [0.06337464909724033, -0.679638896132783, -3.1087325315620733, 2.0882339360702575,
    -1.1201172410014792, -0.031008601325809293, -2.0744261217334135]
  time_from_start: 0.0
- positions: [0.06335930908588873, -0.7354151774072313, -2.939624246421942, 1.8341256735249563,
    -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
  time_from_start: 1.0
- positions: [0.06335930908588873, -0.7231278283145929, -2.9385504456273295, 2.2121050027803877,
    -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
  time_from_start: 2.0
- positions: [0.06335930908588873, -0.7354151774072313, -2.939624246421942, 1.8341256735249563,
    -1.1201355028397157, -0.031008601325809293, -2.0744261217334135]
  time_from_start: 3.0
```

Predefined motions: Action Server

- ROS topics: `rostopic list | grep -i play_motion`

```
/play_motion/cancel
/play_motion/feedback
/play_motion/goal
/play_motion/result
/play_motion/status
```

- **play_motion** action goal API:
  - **motion_name**
    name from the motion definition file
  - **skip_planning**
    when true, skip motion planning in execution
  - **priority**
    unimplemented, will serve to guide preemption policy

- **Preemption policy**
  - While a goal is active, reject all incoming goals
  - Unlike typical action servers (new goal preempts active one)

PAL
ROBOTICS

Predefined motions: Action Clients

joystick
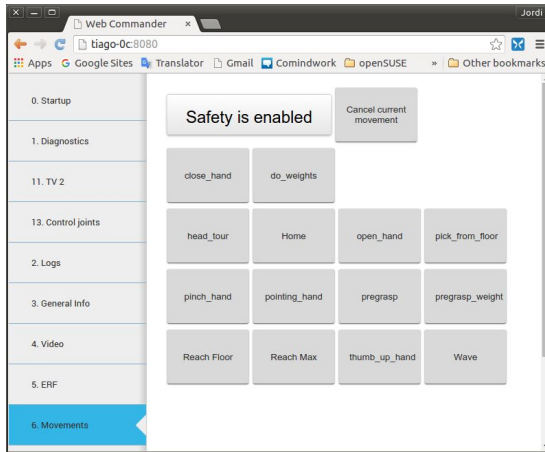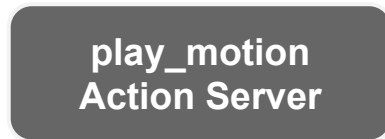
play_motion Action Server

Python Action Client

C++ Action Client

tiago_tutorials

Web commander

play_motion_msgs/PlayMotion GUI Client

Goal
motion_name: 'wave'
skip_planning: False
priority: 0

Feedback

Result

SEND GOAL

CANCEL GOAL

No Goal
Connected to server

rosrun actionlib axclient.py /play_motion

PAL ROBOTICS

TIAGo T.S.
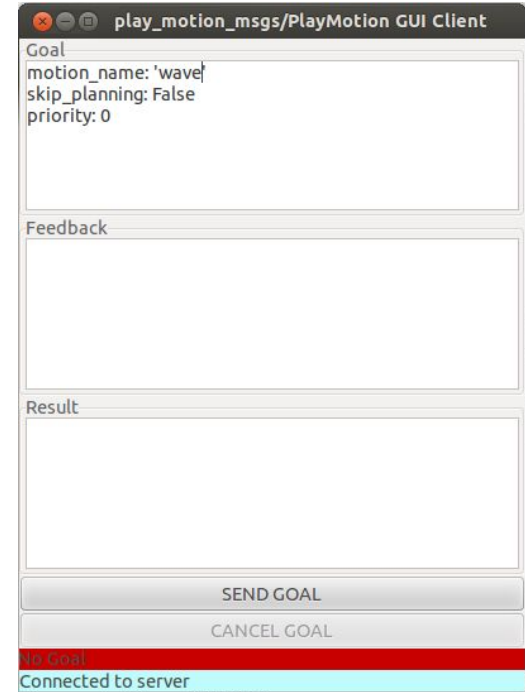
How does play_motion work?

action client

list of involved joints, i.e.:

(torso_lift_joint,
head_1_joint, head_2_joint,
arm_1_joint, arm_2_joint, arm_3_joint, arm_4_joint,
arm_5_joint, arm_6_joint, arm_7_joint,
hand_thumb_joint, hand_index_joint, hand_mrl_joint)

play_motion acts as a demultiplexer to different action servers

play_motion action server

/torso_controller/
follow_joint_trajectory

/head_controller/
follow_joint_trajectory

/arm_controller/
follow_joint_trajectory

/hand_controller/
follow_joint_trajectory

TIAGo T.S.

PAL ROBOTICS

Predefined motions execution (I)

- GUI interface:

your computer IP

ROS_IP=10.68.0.128

rosrun actionlib axclient.py /play_motion

play_motion_msgs/PlayMotion GUI Client

Goal
```
motion_name: 'wave'
skip_planning: False
priority: 0
```
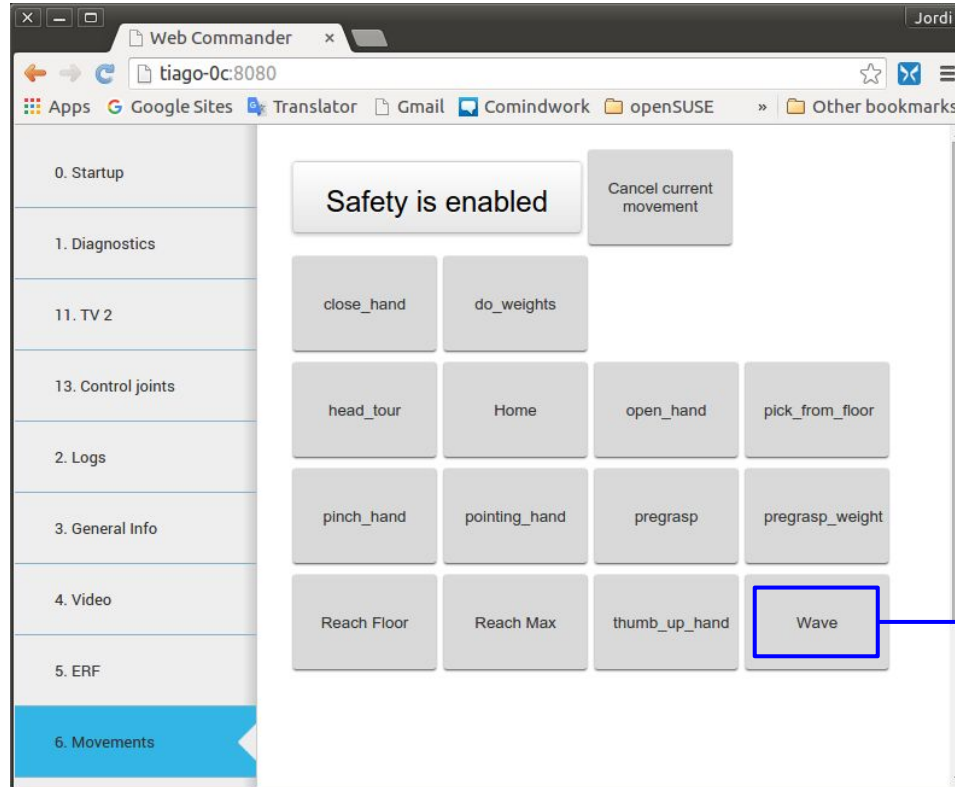
Feedback

Result

SEND GOAL

CANCEL GOAL

Connected to server

2) Fill in the name of the motion

1) Make sure that the client gets connected to the server

3) Press button to send the goal

PAL ROBOTICS

Predefined motions execution (II)

- Web interface

Assigning motions to the Web interface

gedit `rospack find tiago_bringup`/config/tiago_motions.yaml

```yaml
play_motion:
  controllers: [arm_controller, head_controller, torso_controller, hand_controller]
  motions:
    nod:
      joints: [head_1_joint, head_2_joint]
      points:
      - positions: [0.0, 0.0]
        time_from_start: 2.0
      - positions: [0.0, 0.5]
        time_from_start: 3.5
      - positions: [0.0, -0.10]
        time_from_start: 5.5
      - positions: [0.0, 0.0]
        time_from_start: 7.0
      meta:
        name: Nod
        usage: demo
        description: 'nod'
```
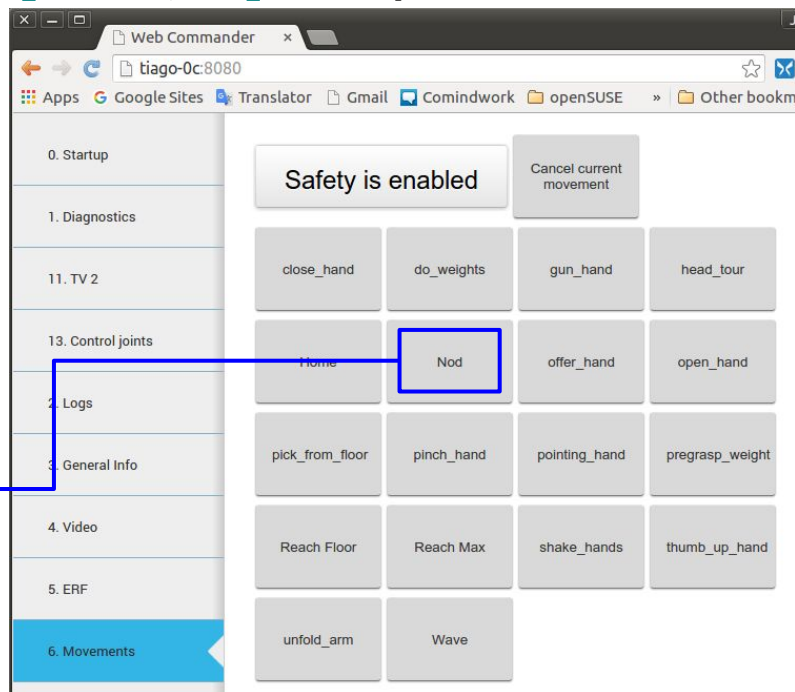
Joystick motion triggers

Joystick upper body motion triggers

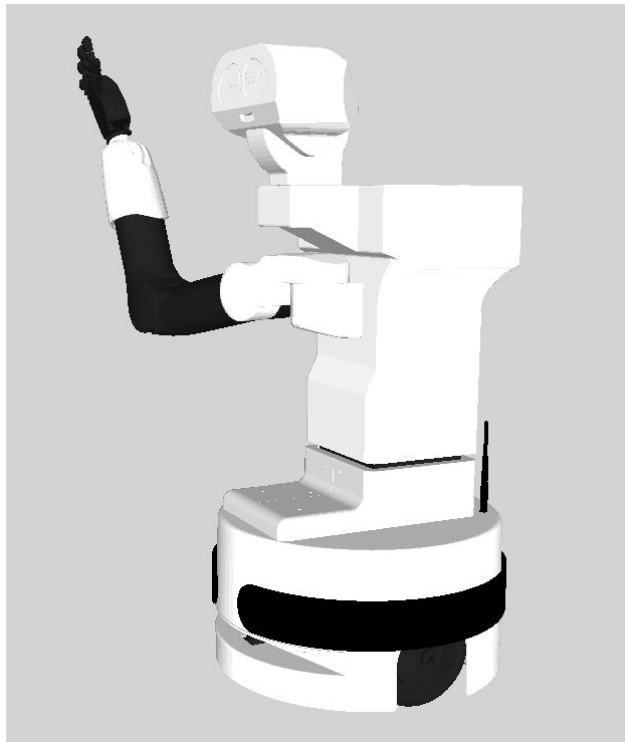● Motion trigger example:

rosparam get /teleop/head_up -p

```
action_goal:
  increment_by: [0.2, 0]
action_name: /head_controller/increment
buttons: [0]
type: action
```

rosparam get /teleop/open_hand -p

```
action_goal:
  motion_name: open_hand
  skip_planning: true
action_name: /play_motion
buttons: [5]
type: action
```

Default joystick triggers location:
tiago_bringup/config/joy_teleop.yaml

PAL ROBOTICS

Questions?