ROS basics
for Aldebaran/SoftBank robots

Natalia Lyubova

12.06.2016

Setup your system

Check the setup doc: https://github.com/nlyubova/tutorials-for-Nao-Pepper-Romeo

- install Ubuntu (14.04 is fine, others are possible), check http://howtoubuntu.org/how-to-install-ubuntu-14-04-trusty-tahr
- install ROS (Indigo is recommended), check http://wiki.ros.org/Installation
  - ```
    sudo apt-get install ros-indigo-desktop
    source /opt/ros/indigo/setup.bash
    ```
- if you use Nao (recommended), check http://wiki.ros.org/nao
  - ```
    sudo apt-get install ros-indigo-nao-robot ros-indigo-nao-meshes
    ```
- if you use Pepper, check http://wiki.ros.org/pepper
  - ```
    sudo apt-get install ros-indigo-pepper-robot ros-indigo-pepper-meshes
    ```
- if you use Romeo, check http://wiki.ros.org/romeo
  - ```
    sudo apt-get install ros-indigo-romeo-robot
    ```

Outline

- Setup your system

1. ROS wiki
2. Basic config
3. Naoqi Driver
4. Visual sensors
5. High-level capabilities

ROS wiki

- http://wiki.ros.org/nao
- http://wiki.ros.org/pepper
- http://wiki.ros.org/romeo

For issues and questions, check ROS SIG Aldebaran

ROS wiki

- http://wiki.ros.org/nao
- http://wiki.ros.org/pepper
- http://wiki.ros.org/romeo

ROS.org About | Support | Status | answers.ros.org Search: Submit

Documentation Browse Software News Download

nao

Aldebaran Nao

Nao is a commercially available humanoid robot built by Aldebaran. The ROS driver was originally developed by Freiburg's Humanoid Robots Lab and Armin Hornung. It essentially wraps the needed parts of Aldebaran's NaoQI API (versions 1.14 and 2.1) and makes it available in ROS. It also provides a complete robot model (URDF).

Robots using ROS: Aldebaran Nao

Robots using ROS: Uni Freiburg's "Osiris" Nao

Wiki
Distributions
ROS/Installation
ROS/Tutorials
RecentChanges
nao

Page
Edit (Text)
Edit (GUI)
Info
Subscribe
Add Link
Attachments
More Actions:

User
NataliaLyubova
Settings
Logout

Contents
1. Aldebaran Nao
   1. Community
   2. Tutorials
   3. Library Overview
      1. Basic Configuration
      2. Hardware Drivers and Simulation
      3. High-Level Capabilities
   4. Simulation

1. Community

There is an official SIG for NAO at https://groups.google.com/forum/?fromgroups#!forum/ros-sig-aldebaran. Please subscribe to it to get the latest news !

2. Tutorials

A complete list of tutorials can be found under tutorials. This includes the installation, startup and further advanced instructions how to connect ROS with your NAO.

- Start all robot nodes: nao_bringup
- See getting started for a walk-through guide to installing ROS, NAOqi, and rviz (may be outdated by now).

3. Library Overview

The core functionality is implemented in the nao_robot stack (can be installed on the robot or on a remote PC), extended with further functionality in nao_extras (should be installed on a remote PC).

```
sudo apt-get install ros-.*-nao-robot
sudo apt-get install ros-.*-nao-extras
```

For an outline of the libraries included, please see the tables below.

3.1 Basic Configuration

Capability	ROS package/stack
Robot-specific Messages and Services	naoqi_bridge_msgs
Robot model (URDF)	nao_description
Robot meshes	nao_meshes

3.2 Hardware Drivers and Simulation

3.3 High-Level Capabilities

Component	ROS package/stack
Teleop	nao_teleop
Footstep planning	footstep_planner
Execute / manage body poses	naoqi_pose
Follow 2D path / walk to target	nao_path_follower
Diagnostics / Visualization	naoqi_dashboard
Interaction	nao_interaction

Outline

- Setup your system

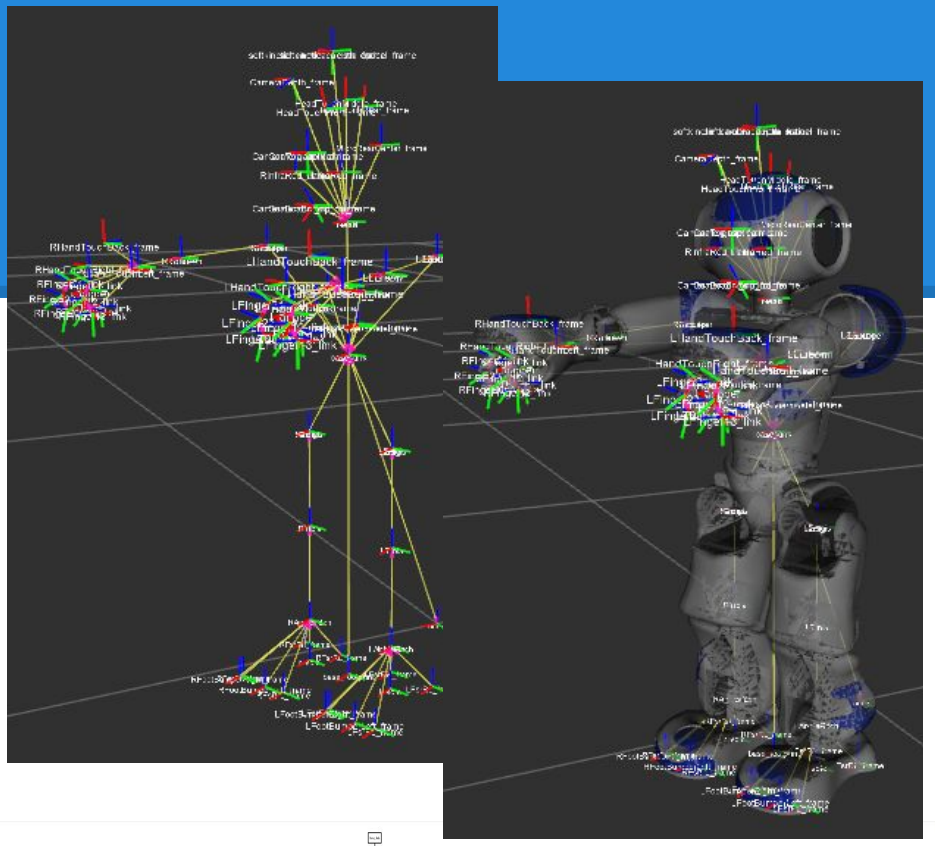
1. ROS wiki
2. Basic config
3. Naoqi Driver
4. Visual sensors
5. High-level capabilities

1. Basic config

nao_robot package

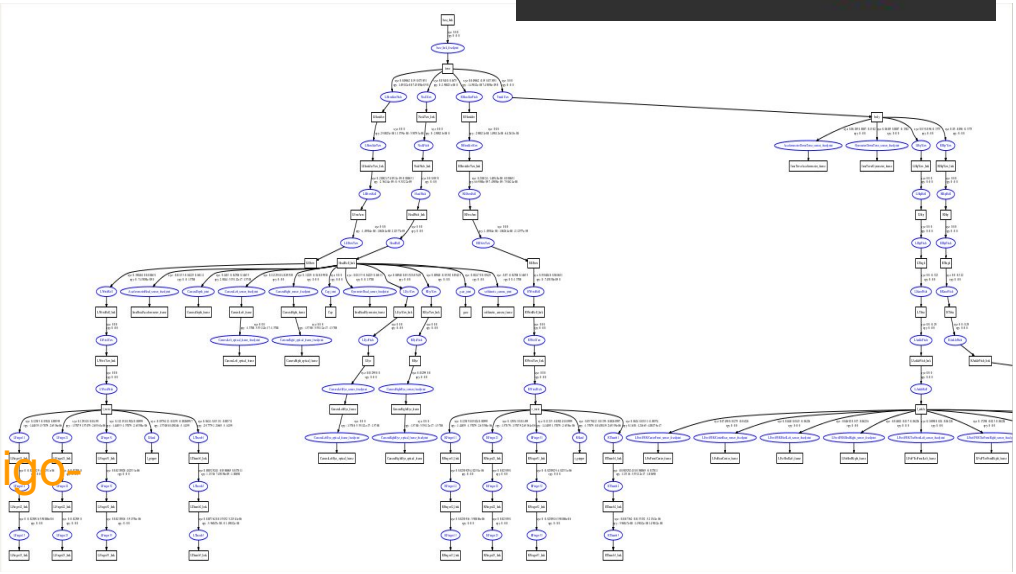
- source code:
  - https://github.com/ros-naoqi/nao_robot
- is a meta-package that includes
  - Robot model (URDF): nao_description
  - Robot-specific Startup: nao_bringup
  - Applications: nao_apps
- install:
  - sudo apt-get install ros-indigo-nao-robot

1. Basic config robot description



nao_robot/nao_description package

- http://wiki.ros.org/nao_description
- includes:
  - URDF (Universal Robot Description Format): XML describing the rigid kinematics
  - meshes
  - basis for
    - low level: kinematics, actuators/sensors
    - high level: planning, navigation, grasping
    - GUIs
- install: sudo apt-get install ros-indigo-nao-description



Outline

- Setup your system

1. ROS wiki
2. Basic config
3. Naoqi Driver
4. Visual sensors
5. High-level capabilities

2. Naoqi driver

naoqi_driver package

- Driver module between NAOqi OS and ROS
- Common for all Aldebaran robots
- Doc: http://ros-naoqi.github.io/naoqi_driver
- Gets data from NAOqi hence ensuring low latency and CPU usage
  - actuator data
  - sensor data
  - basic diagnostic for battery, temperature
- Allows to control:
  - /cmd_vel
  - /move_base_simple/goal
  - /joint_angles
- You can run it locally on a robot or remotely on PC

2. Naoqi driver

naoqi_driver package

Install: http://ros-naoqi.github.io/naoqi_driver/install.html

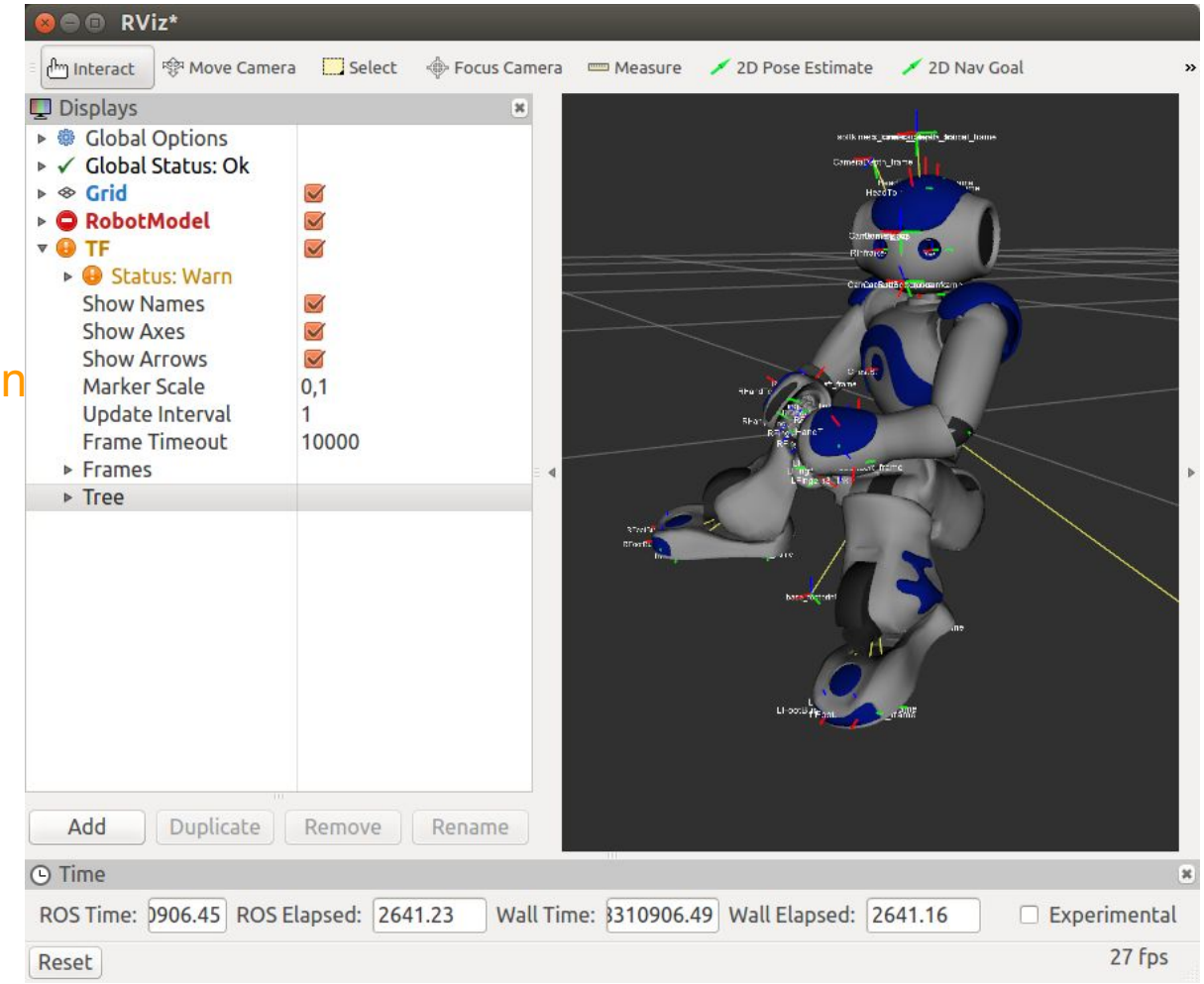
- install it from the official release
  - sudo apt-get install ros-indigo-naoqi-driver
- or compile it from source:
  - Download C++ NaoqiSDK and Python NaoqiSDK
    - create an account and login in https://community.ald.softbankrobotics.com
    - go to resources, download C++ NaoqiSDK, extract it
  - mkdir -p ~/catkin_ws/src && cd ~/catkin_ws/src
  - git clone https://github.com/ros-naoqi/naoqi_driver
  - # make sure you get all the dependencies installed
  - rosdep install -i -y --from-paths ./naoqi_driver
  - source /opt/ros/indigo/setup.sh
  - cd ../ && catkin_make

2. Naoqi driver

launch

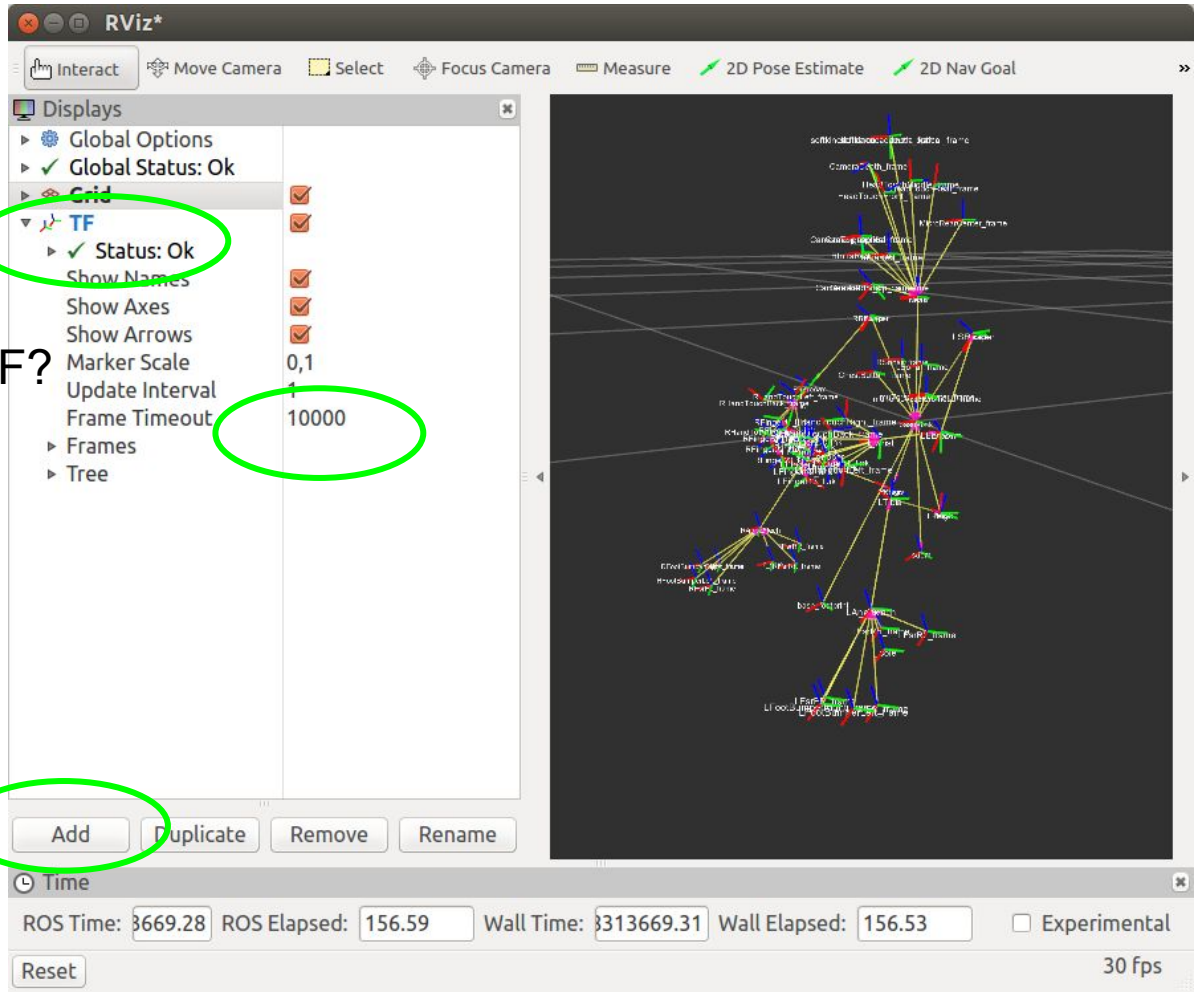
- roslaunch naoqi_driver naoqi_driver.launch nao_ip:=<robot_IP> roscore_ip:=<roscore_ip> network_interface:=<eth0|wlan0|tethering|vpn>

- Example of launch: roslaunch naoqi_driver naoqi_driver.launch nao_ip:=10.1.0.102 network_interface:=wlan0



2. Naoqi driver

- Start RviZ
  - rosrun rviz rviz
- add Plugins
  - TF
- check
  - do you see robot's TF?
- set TF params:
  - Update Interval
  - FrameTimeout



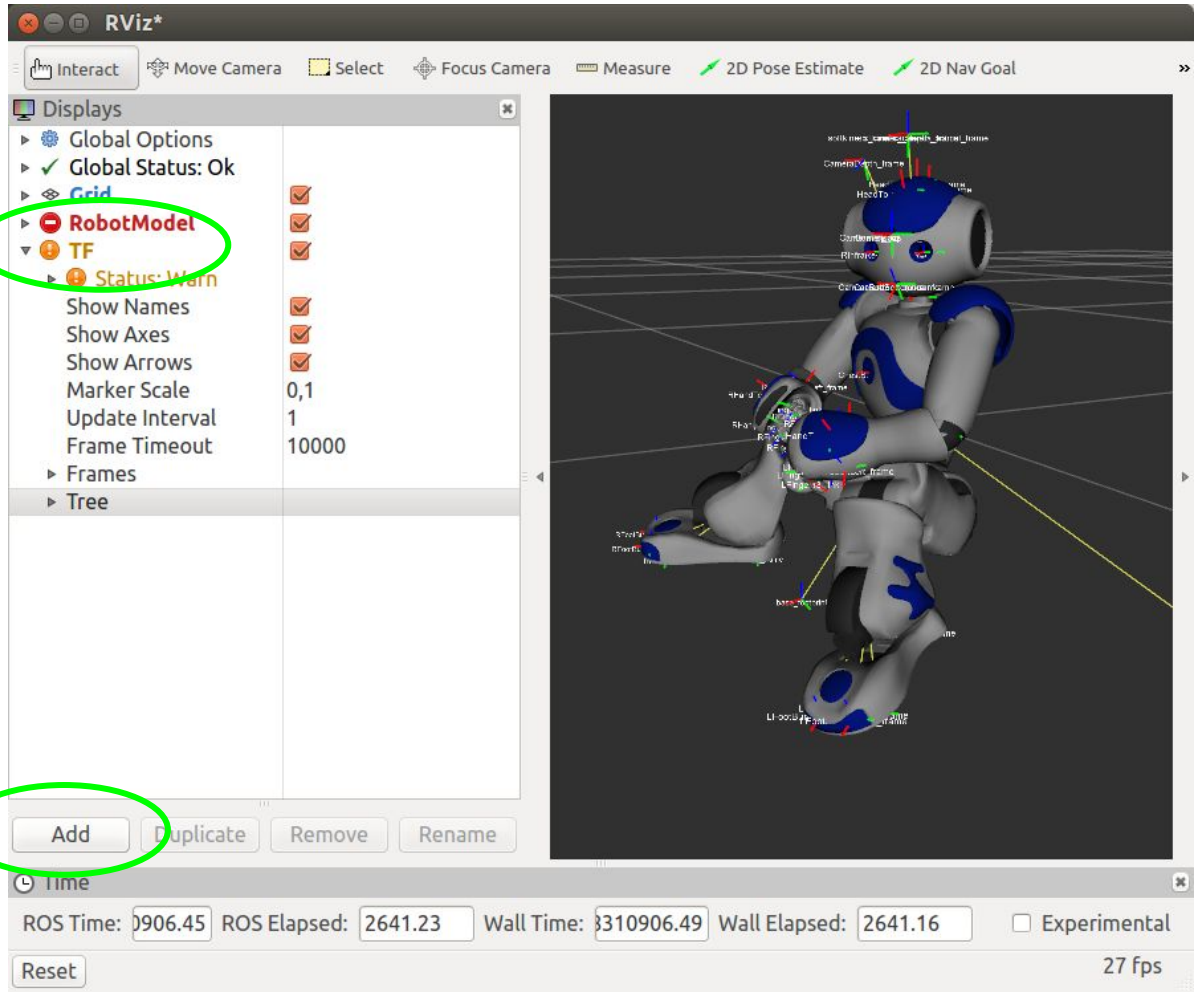
2. Naoqi driver

In RViZ:

- add Plugins
  - RobotModel
- check
  - do you see a robot?

Check ROS topics:

- rostopic list
- rostopic echo /tf



2. Naoqi driver

If you have problems when connecting to your robot, set up the following environment variables, to find the robot and rosmaster:

- your robot's IP
- export NAO_IP=10.1.0.102

- optionally, the address of a ROS Node/tool
- export ROS_IP=<IP of your PC>
- export ROS_HOSTNAME=$ROS_IP

- rosmaster IP; set to the XML-RPC URI of the master
- export ROS_MASTER_URI=http://$ROS_IP:11311

2. Naoqi driver

rostopic list

/audio
/camera/bottom/camera_info
/camera/bottom/image_raw
/camera/bottom/image_raw/compressed
/camera/bottom/image_raw/compressed/parameter_descriptions
/camera/bottom/image_raw/compressed/parameter_updates
/camera/bottom/image_raw/theora
/camera/bottom/image_raw/theora/parameter_descriptions
/camera/bottom/image_raw/theora/parameter_updates
/camera/front/camera_info
/camera/front/image_raw
/camera/front/image_raw/compressed
/camera/front/image_raw/compressed/parameter_descriptions
/camera/front/image_raw/compressed/parameter_updates
/camera/front/image_raw/theora
/camera/front/image_raw/theora/parameter_descriptions
/camera/front/image_raw/theora/parameter_updates
/clicked_point
/cmd_vel
/diagnostics
/imu/torso
/info
/initialpose
/joint_angles
/joint_states
/move_base_simple/goal
/rosout
/rosout_agg
/sonar/left
/sonar/right
/tf
/tf_static

2. Naoqi driver

Check it out:

- check ROS topics
  - rostopic list
  - rostopic echo /tf
  - rostopic echo /joint_states
  - rostopic echo /camera/front/camera_info
- check tf
  - rosrun tf tf_monitor
  - rosrun tf tf_echo <frame1> <frame2>
- record data and play them
  - rosbag record -a
  - rosbag play yourbagname.bag

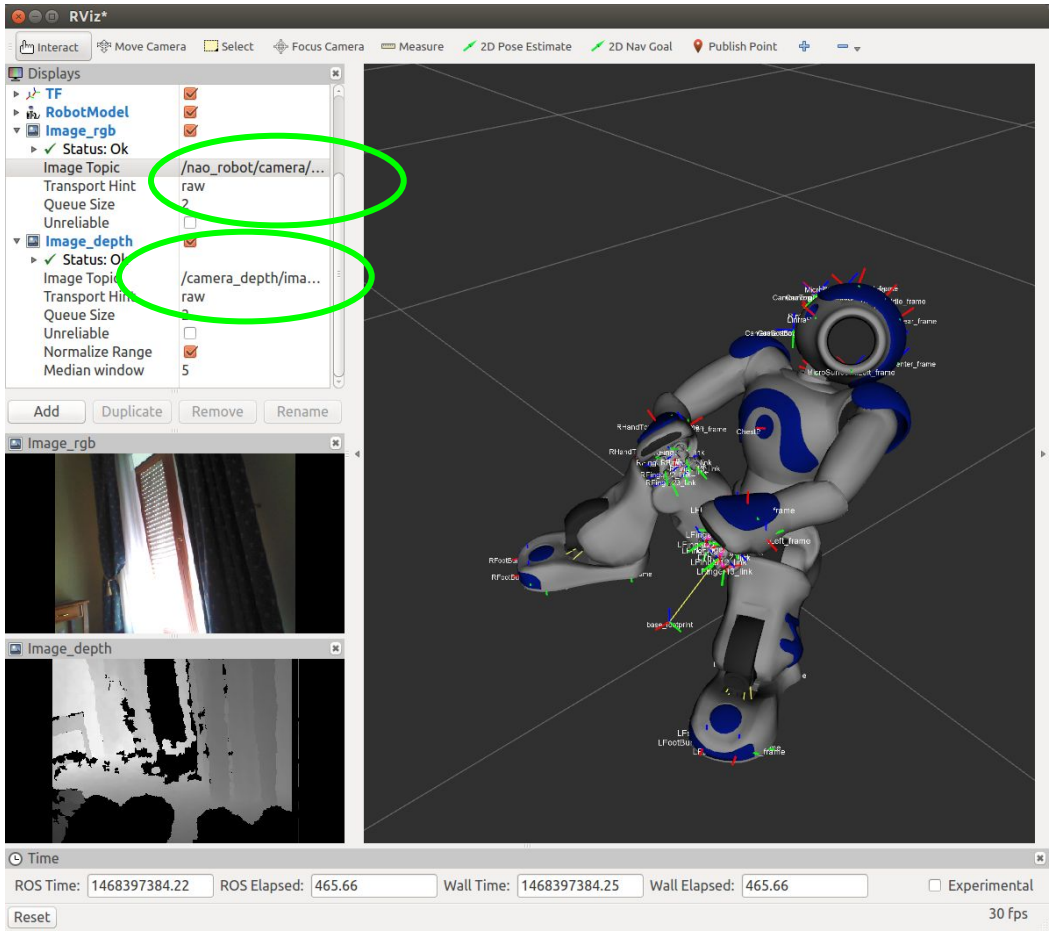
2. Naoqi driver

In RViZ:

- add Plugins
  - Image
- select the topic /nao_robot/camera/front /image_raw
- do you see RGB image from the robot?

Try to add plugins:

- PointCloud2
- DepthCloud



2. Naoqi driver

How to get a colored pointCloud ?

1) Pointcloud registration:

- rosrun nodelet nodelet manager __name:=nodelet_manager
- rosrun nodelet nodelet load depth_image_proc/register nodelet_manager rgb/camera_info:=/naoqi_driver_node/camera/front/camera_info depth/camera_info: =/naoqi_driver_node/camera/depth/camera_info depth/image_rect: =/naoqi_driver_node/camera/depth/image_raw depth_registered/image_rect: =/naoqi_driver_node/camera/depth_registered/image_rect

2) RGB pointcloud:

- rosrun nodelet nodelet load depth_image_proc/point_cloud_xyzrgb nodelet_manager rgb/camera_info:=/naoqi_driver_node/camera/front/camera_info rgb/image_rect_color: =/naoqi_driver_node/camera/front/image_raw depth_registered/image_rect: =/naoqi_driver_node/camera/depth_registered/image_rect depth_registered/points: =/naoqi_driver_node/camera/depth_registered/points

Outline

- Setup your system

1. ROS wiki
2. Basic config
3. Naoqi Driver
4. Visual sensors
5. High-level capabilities

3. Visual sensors

All sensors supported in ROS: http://wiki.ros.org/Sensors

We work with:

- ASUS Xtion or Kinect sensor
  - install: sudo apt-get install ros-indigo-openni-launch
  - launch: roslaunch openni_launch openni.launch depth_frame_id:=/CameraDepth_frame rgb_frame_id:=/CameraDepth_frame
  - for image registration: rosrun rqt_reconfigure rqt_reconfigure
- Creative senz3d
  - install: git clone https://github.com/nlyubova/softkinetic
  - launch: roslaunch softkinetic_camera softkinetic_camera_simple.launch
- Intel SR200 and F200
  - install: git clone https://github.com/nlyubova/realsense
  - launch: roslaunch realsense_camera realsense_sr300.launch
- Intel R200
  - install: git clone https://github.com/intel-ros/realsense
  - launch: roslaunch realsense_camera realsense_r200_rgbd.launch

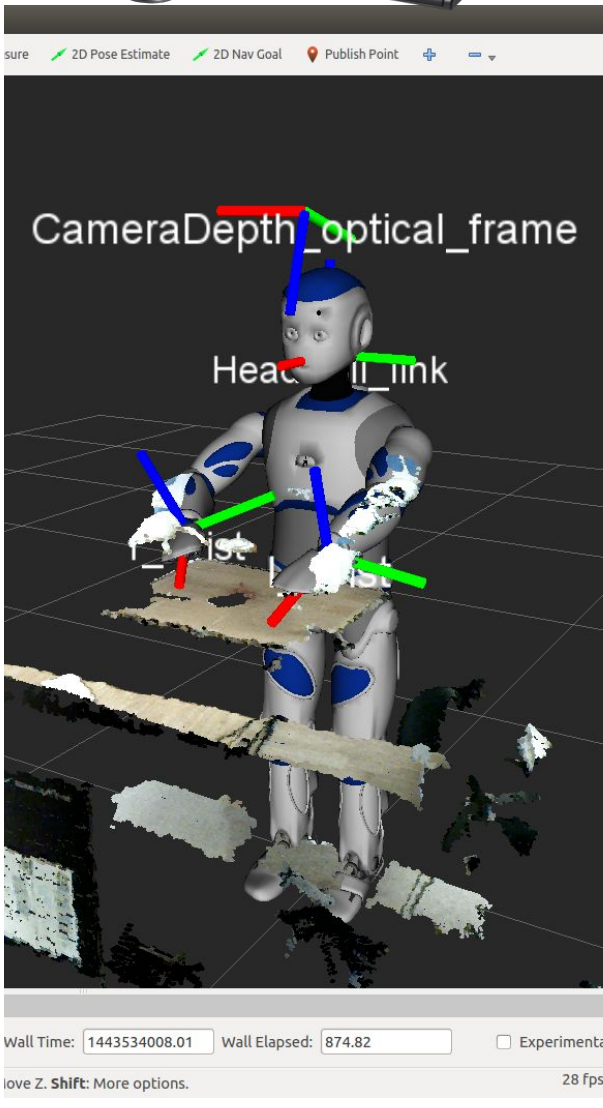
3. Visual sensors

Adding an external sensor:

- add a reference frame to URDF
  - position/orientation
  - child/parent links
- check the sensor's position/orientation in RviZ

Example: here

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
        <link name="CameraF200_frame"/>
        <joint name="CameraF200_joint" type="fixed">
                <parent link="HeadRoll_link"/>
                <child link="CameraF200_frame"/>
                <origin rpy="0.0 0.0 0.0" xyz="0.14 0.0 0.185"/>
        </joint>
        <link name="CameraF200_depth_optical_frame"/>
        <joint name="CameraF200_depth_optical_frame_fixedjoint" type="fixed">
                <parent link="CameraF200_frame"/>
                <child link="CameraF200_depth_optical_frame"/>
                <origin rpy="-2.7 0 -1.5708" xyz="0.0 0.0 0.0"/>
        </joint>
        <link name="CameraF200_uncalib_depth_optical_frame"/>
        <joint name="CameraF200_uncalib_depth_optical_frame_fixedjoint" type="fixed">
                <parent link="CameraF200_frame"/>
                <child link="CameraF200_uncalib_depth_optical_frame"/>
                <origin rpy="-2.61799 0 -1.5708" xyz="0.0 0.0 0.0"/>
        </joint>
</robot>
```



Outline

- Setup your system

1. ROS wiki
2. Basic config
3. Naoqi Driver
4. Visual sensors
5. High-level capabilities

- object recognition
- nao teleop

3. High-level capabilities
Object recognition

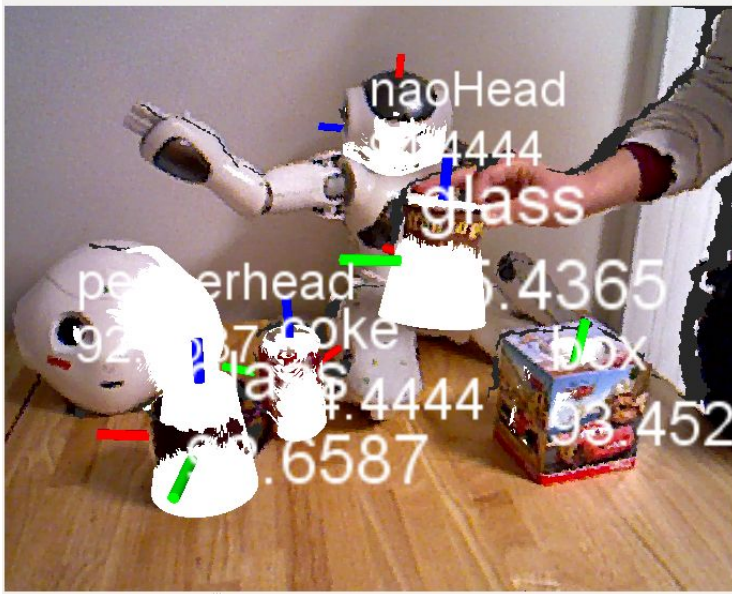
Object Recognition Kitchen ORK:

Doc: http://wg-perception.github.io/object_recognition_core

- object DB management
- i/o handling
- ecto-based computations

  organized as directed acyclic graphs
- several packages recognizing objects

  in different ways:
  - object_recognition_tabletop
  - object_recognition_linemod
  - object_recognition_transparent
  - object_recognition_tod
- Install:



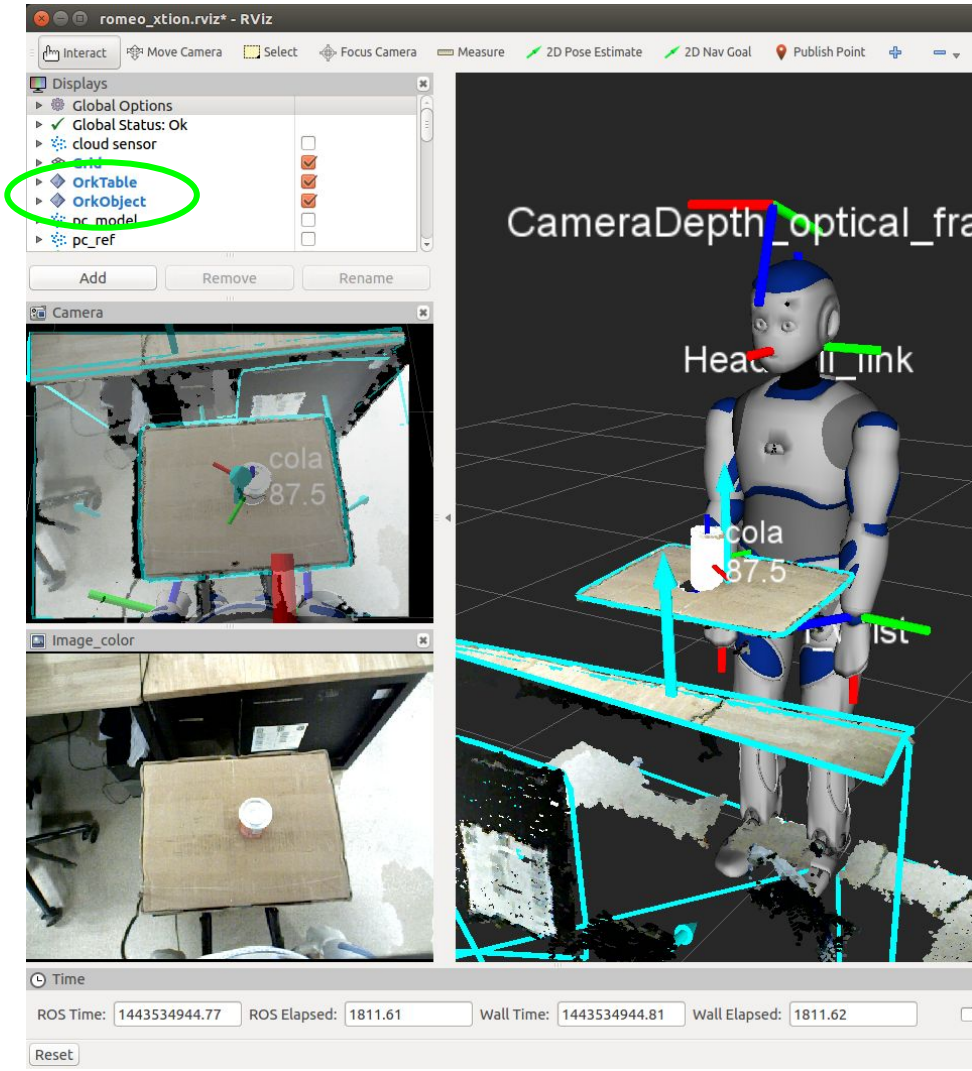
$ sudo apt-get install ros-indigo-object-recognition-*

3. High-level capabilities
Object recognition

- Set the params in detection.ros.ork:
  - rgb/depth_frame_id,
  - rgb/depth_image_topic,
  - rgb/depth_camera_info
- Launch continuous detection:
  - rosrun object_recognition_core detection -c detection.ros.ork
- Or Launch server+client:
  - rosrun object_recognition_ros server -c detection.ros.ork
  - rosrun object_recognition_ros client

See the outcome in RviZ with Plugins:

- OrkObject
- OrkTable



3. tele-operating your robot

- nao_teleop package
- Allows to control the robot with a gamepad
  - move the base, rotate
  - move the head, rotate
- is similar to Android application “Teleop”
- Doc: http://wiki.ros.org/nao_teleop
- To launch:
  - export NAO_IP=10.0.160.35
  - roslaunch naoqi_driver naoqi_driver.launch nao_ip:=${NAO_IP} network_interface:=wlan0
  - roslaunch nao_teleop teleop_joy.launch

Packages described in this tutorial:

1. Basic config: meta-package: nao_robot
   - Robot model (URDF): nao_description
   - Robot-specific Startup: nao_bringup
2. Naoqi Bridge
   - driver : naoqi_driver
   - depth camera launch: romeo_sensors_py
3. High-level capabilities
   - object recognition ORK

Thank you!