Development of intelligent systems (RInS)

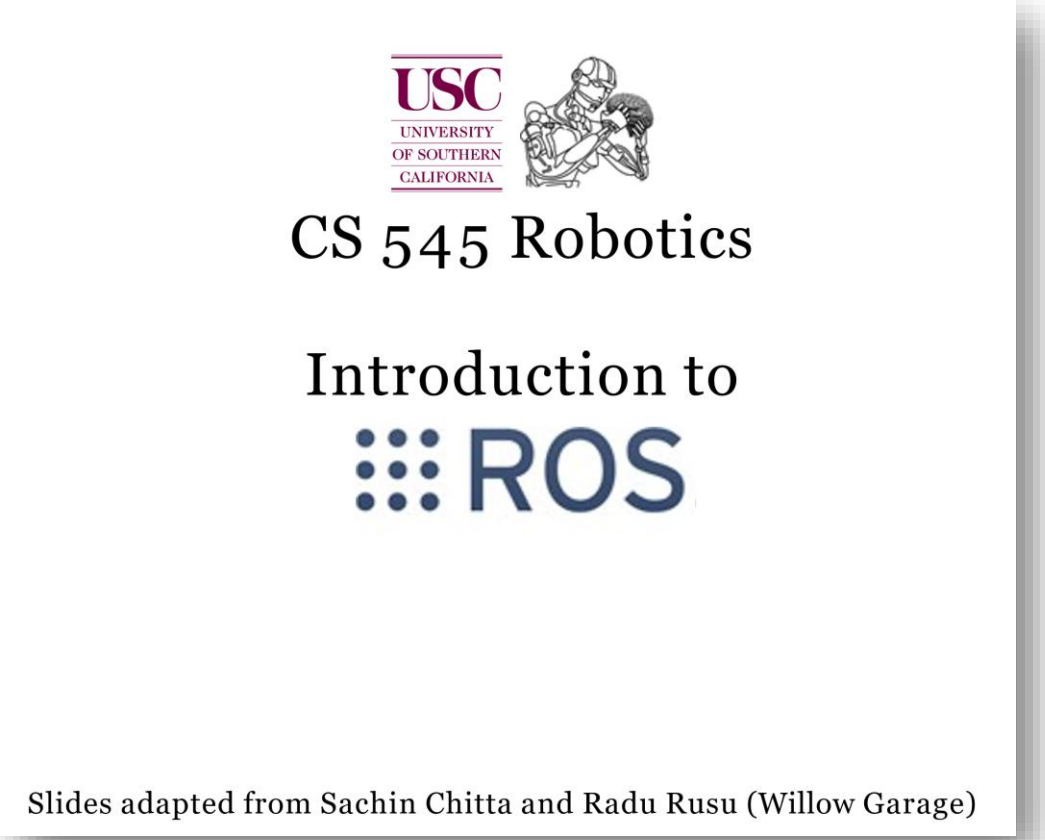
ROS - Robot Operating System

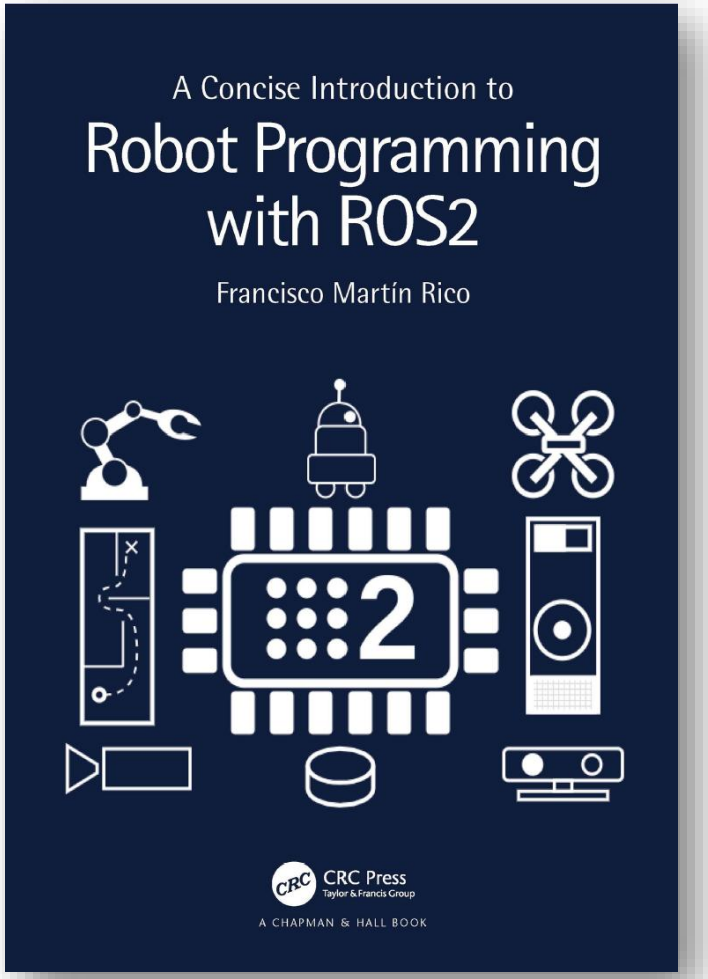
Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

Academic year: 2024/25

ROS, ROS2 – Meta operating System

Slide credits





https://github.com/fmrico/book_ros2

Overview

?

OS

my new application

web browser

email client

window manager

memory management

process management

scheduler

device drivers

file system

Overview

Standards

Hardware: PCI bus, USB port, FireWire, ...

Software: HTML, JPG, TCP/IP, POSIX, ...

my new application

web browser

email client

window manager

memory management

process management

scheduler

device drivers

file system

OS

Overview

...but what about robots ?

my new application

web browser

email client

window manager

memory management

process management

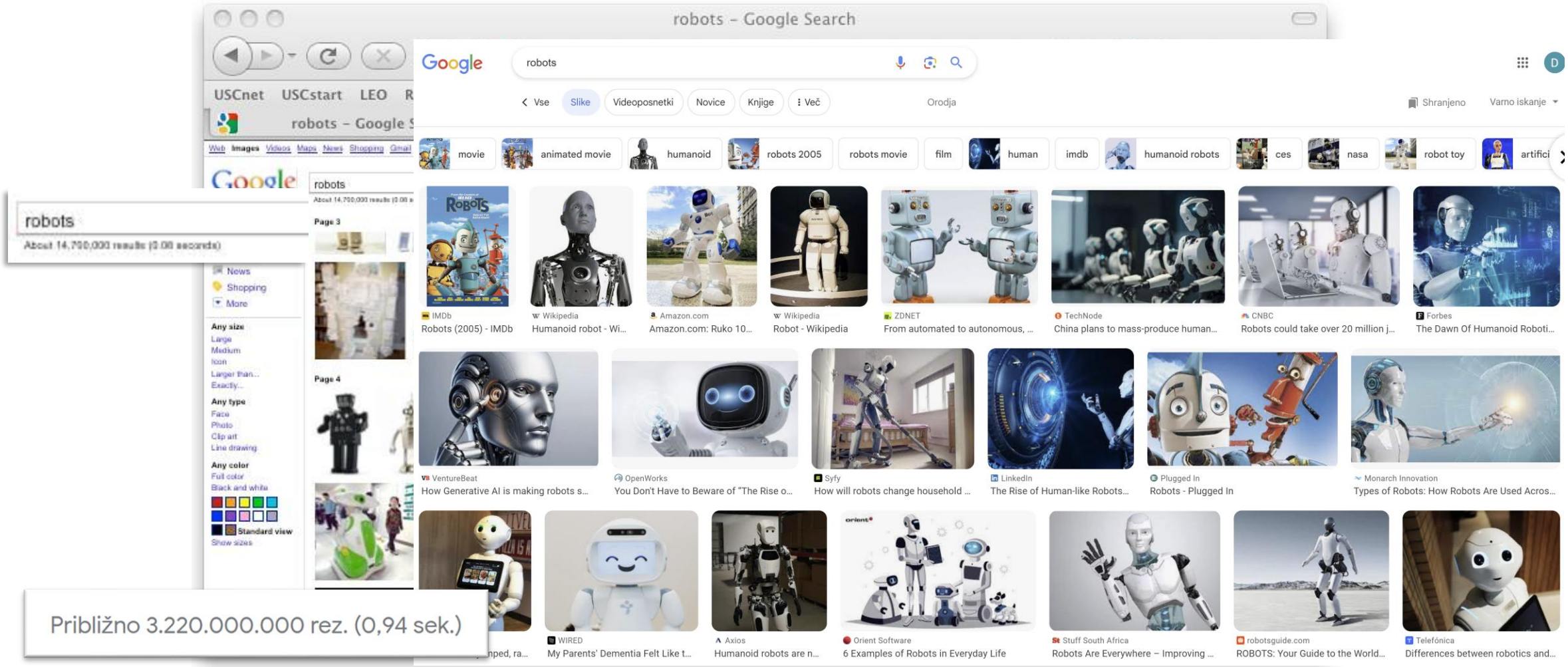
scheduler

device drivers

file system

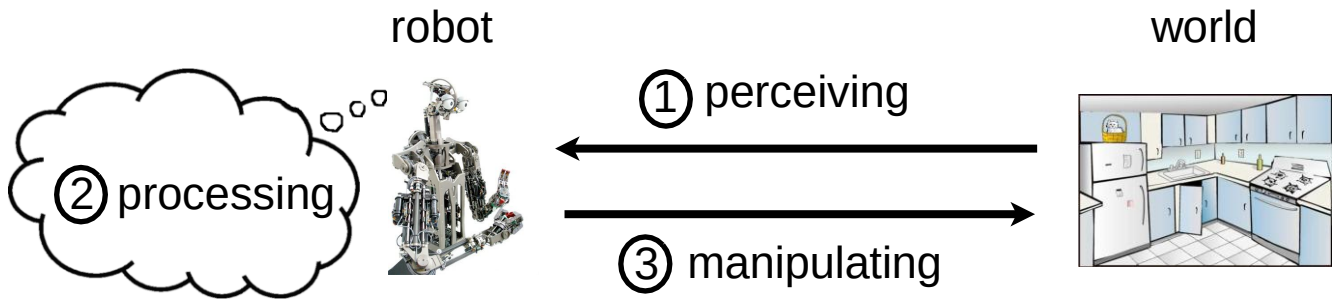
OS

Lack of standards for robotics



USC UNIVERSITY OF SOUTHERN CALIFORNIA

Typical scenario



① Many sensors require device drivers and calibration procedures

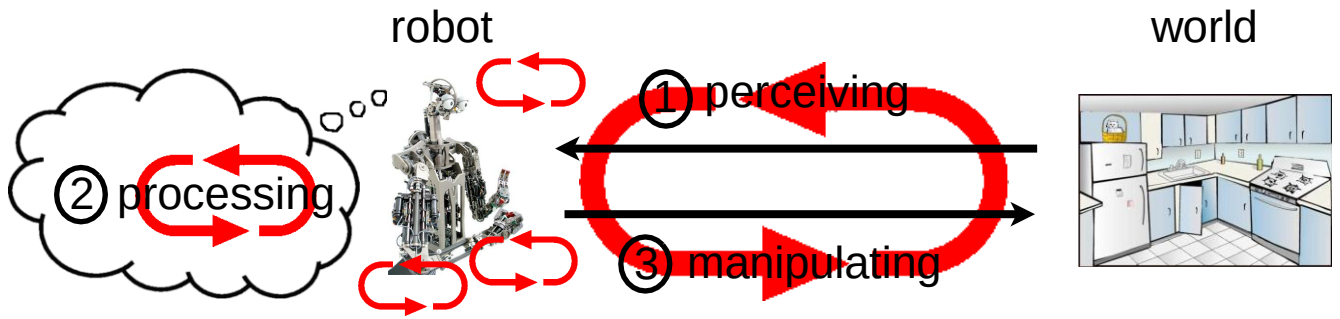
For example cameras: stereo processing, point cloud generation...

Common to many sensors: filtering, estimation, coordinate transformation, representations, voxel grid/point cloud processing, sensor fusion,...

② Algorithms for object detection/recognition, localization, navigation, path/motion planning, decision making, ...

③ Motor control: inverse kinematics/dynamics, PID control, force control, ...

Control loops



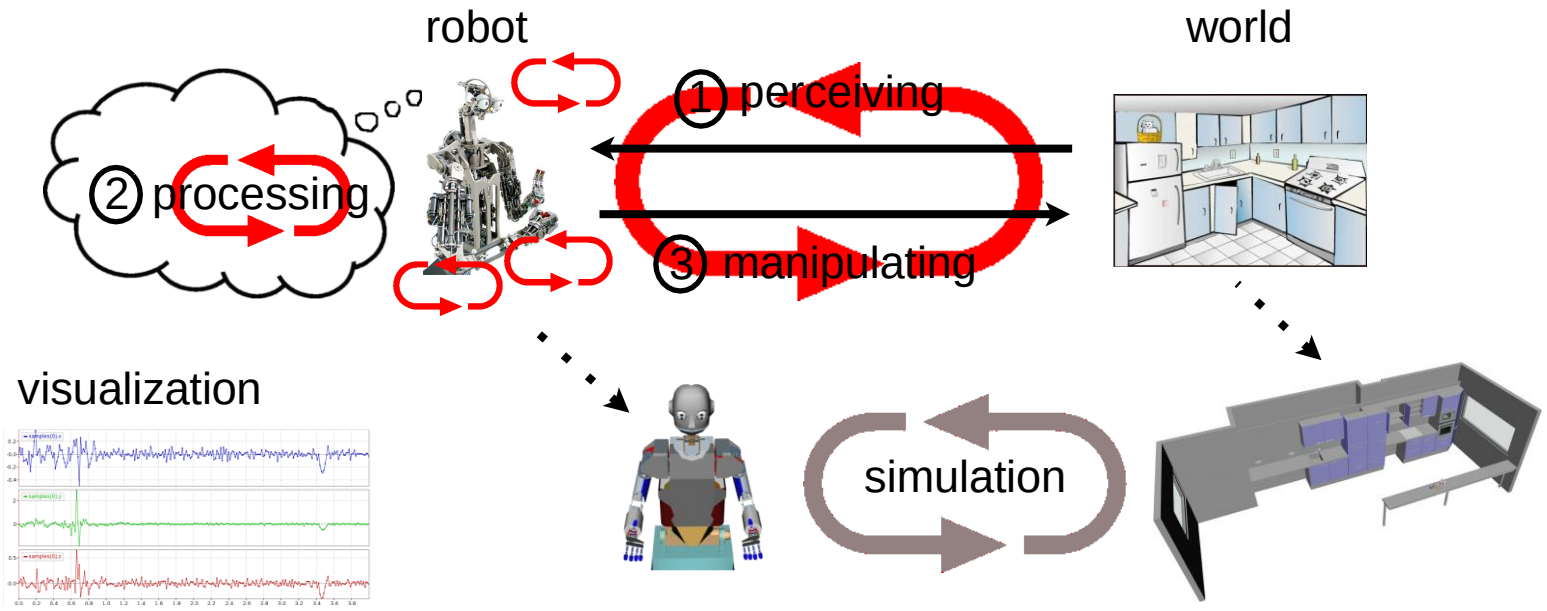
Many control loop on different time scales

- Outer most **control loop** may run once every second (1Hz) or slower
- Inner most may run at 1000Hz or even higher rates

Software requirements:

- Distributed processing with loose coupling. Sensor data comes in at **various time scales**.
- Real time capabilities for tight motor control loops.

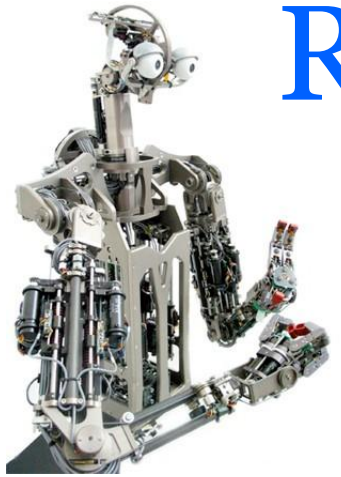
Debugging tools



Simulation: No risk of breaking real robots, reduce debugging cycles, test in super real-time, controlled physics, perfect model is available...

Visualization: Facilitates debugging, ...looking at the world from the robot's perspective. Data trace inspections allow debugging on small time scales.

Overview



ROS

navigation

task executive

visualization

simulation

perception

control

planning

data logging

message passing

device drivers

real-time capabilities

OS

web browser

email client

window manager

memory management

process management

scheduler

device drivers

file system

Overview

1 Orocos: <http://www.orocos.org>
2 OpenRTM: <http://www.is.aist.go.jp>
3 ROS: <http://www.ros.org>
4 OPRoS: <http://opros.or.kr>
5 JOSER: <http://www.joser.org>
6 InterModalics: <http://intermodalics.eu>
7 Denx: <http://denx.de>
8 GearBox: <http://gearbox.sourceforge.net/gbx_doc_overview.html>

Why should we agree on one standard ?

Code reuse, code sharing:

stop inventing the wheel again and again... instead build on top of each other's code.

Ability to run the same code across multiple robots:

portability facilitates collaborations and allows for comparison of similar approaches which is very important especially in science.

What is ROS ?

ROS is an **open-source**, **meta-operating** system and stands for Robot Operating System.

It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

Willow Garage

http://www.ros.org (documentation)

https://lists.sourceforge.net/lists/listinfo/ros-users (mailing list)

http://www.ros.org/wiki/ROS/Installation (it's open, it's free !!)

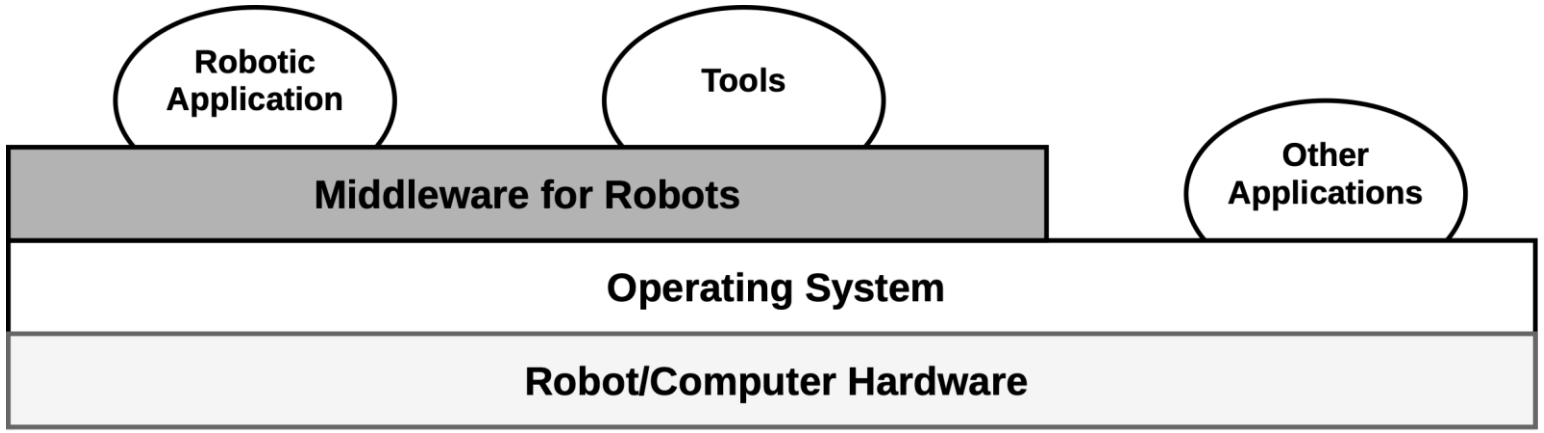
Mainly supported for Ubuntu linux, experimental for Mac OS X and other unix systems.

http://www.ros.org/wiki/ROS/StartGuide (tutorials)

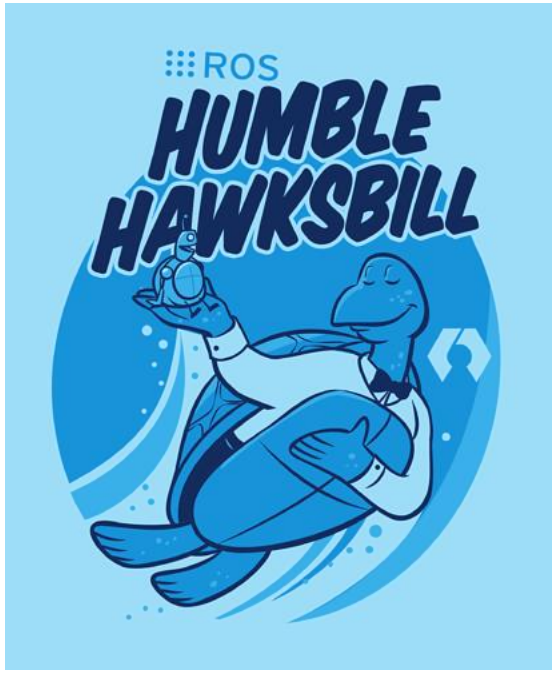
USC UNIVERSITY OF SOUTHERN CALIFORNIA

Programming Robots

- Robots must be programmed to be useful
- We need Middlewares
- Robot programming middlewares provide drivers, libraries, and methodologies
- Few of them have survived the robot for which they were designed or have expanded from the laboratories where they were implemented
- The big difference is the ROS developers community around the world.



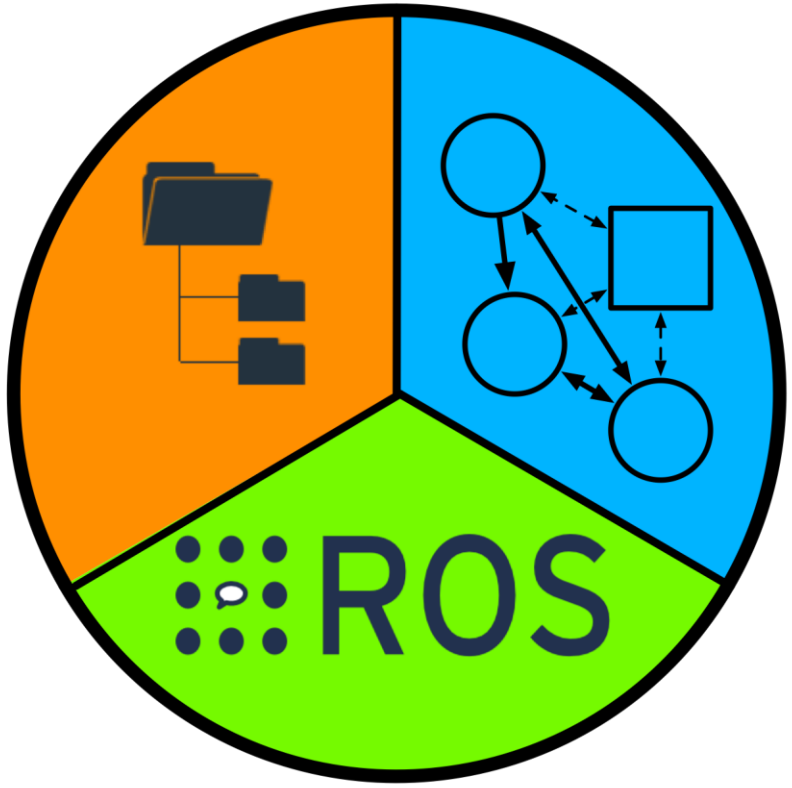
- The acronym ROS is Robot Operating System
- ROS and ROS2
- Lot of tutorials and documentation
- We will use Ubuntu 22.04 + Humble



Intelligent Robotics Lab

ROS Dimensions

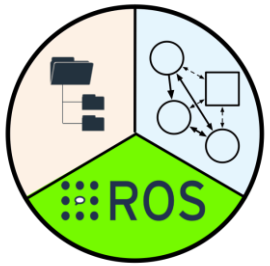
Workspace: the set of software installed on the robot or computer, the programs that the user develops, and tools to build



Computation Graph: a running ROS2 application

Community: vast community of developers who con- tribute with their own applications and utilities through public repositories, to which other developers can contribute

The Community



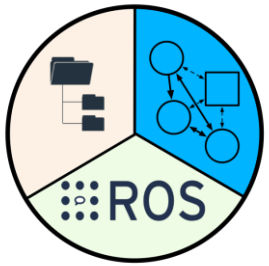
- Open Source and Licenses
- ROS2 organizes software development in federal model
- Packages and distributions
- Online resources

The Workspace

- Approaches ROS2 software from a static point of view.
- Where the ROS2 software is installed, organized, and all the tools and processes that allow us to launch a computing graph.
- This includes the build system and node startup tools.
- Elements:
  - **Package**:
    - It is the minimum functional set of software.
    - Contains executables, libraries, or message definitions with a common purpose.
  - **Workspace**:
    - A directory that contains packages.
    - Activable to be available to use.

Intelligent Robotics Lab

The Computation Graph



- A robot's software looks like during its execution.
- A Computation Graph contains ROS2 nodes that communicate with each other so that the robot can carry out some tasks.
- The logic of the application is in the nodes, as the primary elements of execution in ROS2.
- Communication mechanisms:
  - **Publication/Subscription**: Asynchronous N:M
  - **Services**: Synchronous 1:1
  - **Actions**: Asynchronous 1:1

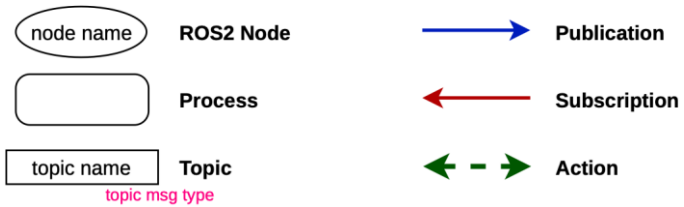
Intelligent Robotics Lab

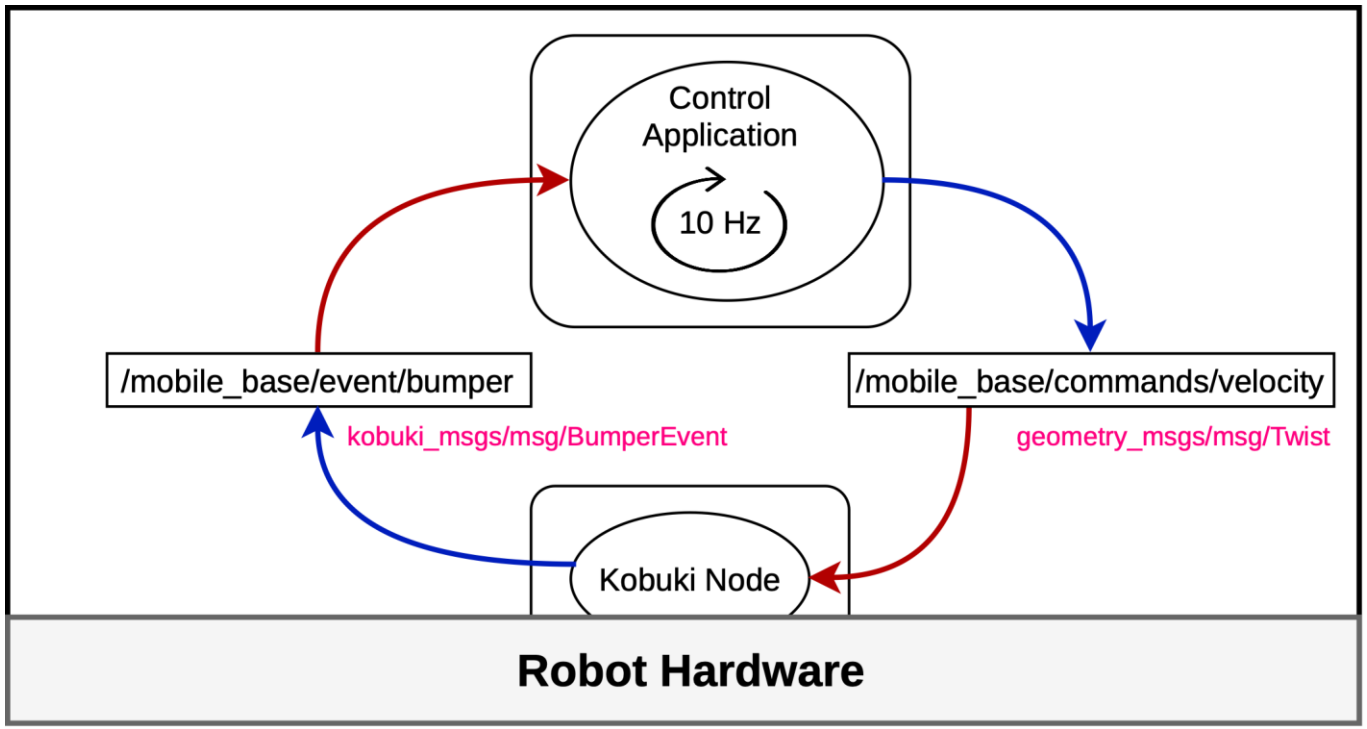
The Computation Graph

- A robot's software looks like during its execution.
- A Computation Graph contains ROS2 nodes that communicate with each other so that the robot can carry out some tasks.
- The logic of the application is in the nodes, as the primary elements of execution in ROS2.

  - Communication mechanisms:
    - **Publication/Subscription**: Asynchronous N:M
    - **Services**: Synchronous 1:1
    - **Actions**: Asynchronous 1:1

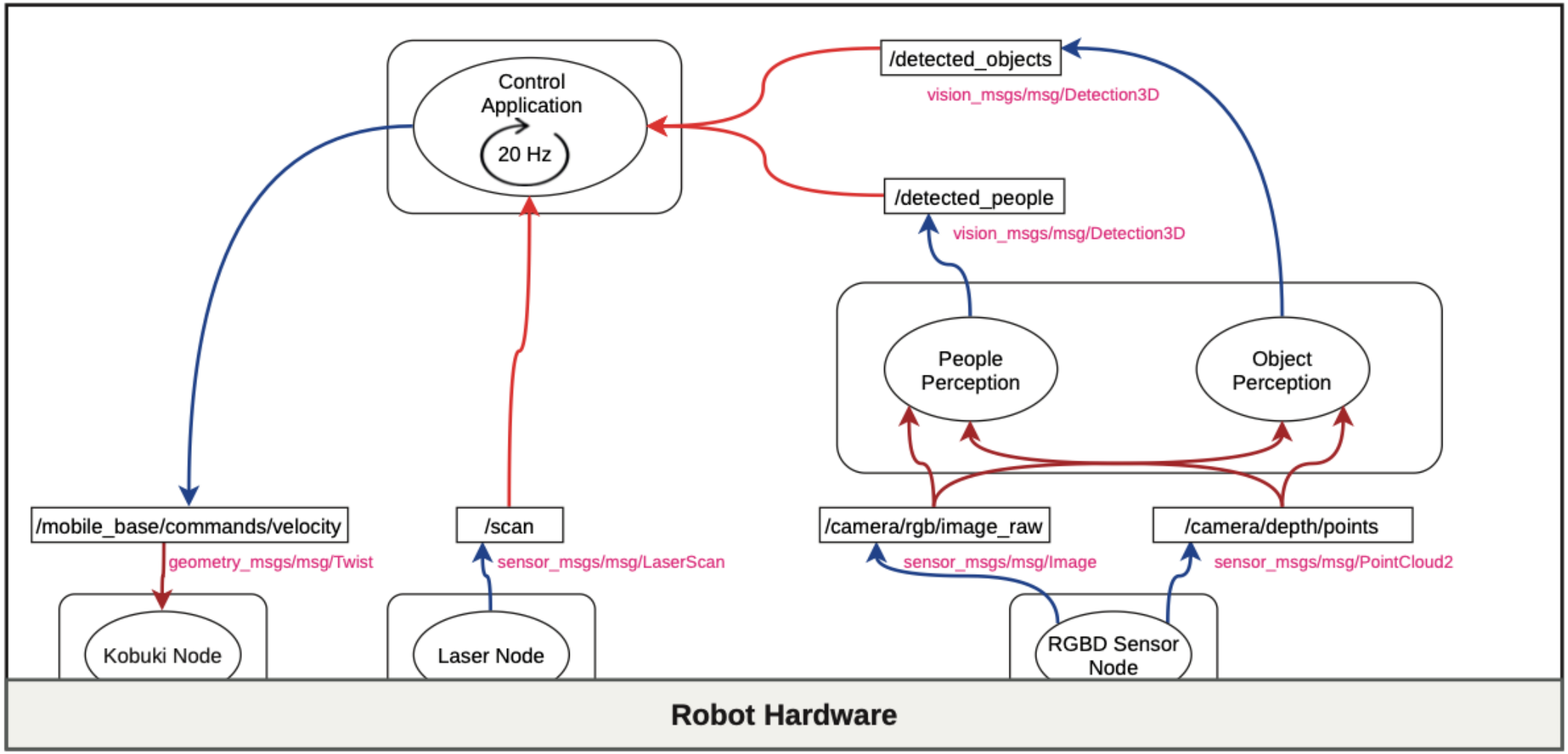
  - Execution model
    - **Iterative**
    - **Event-Oriented**





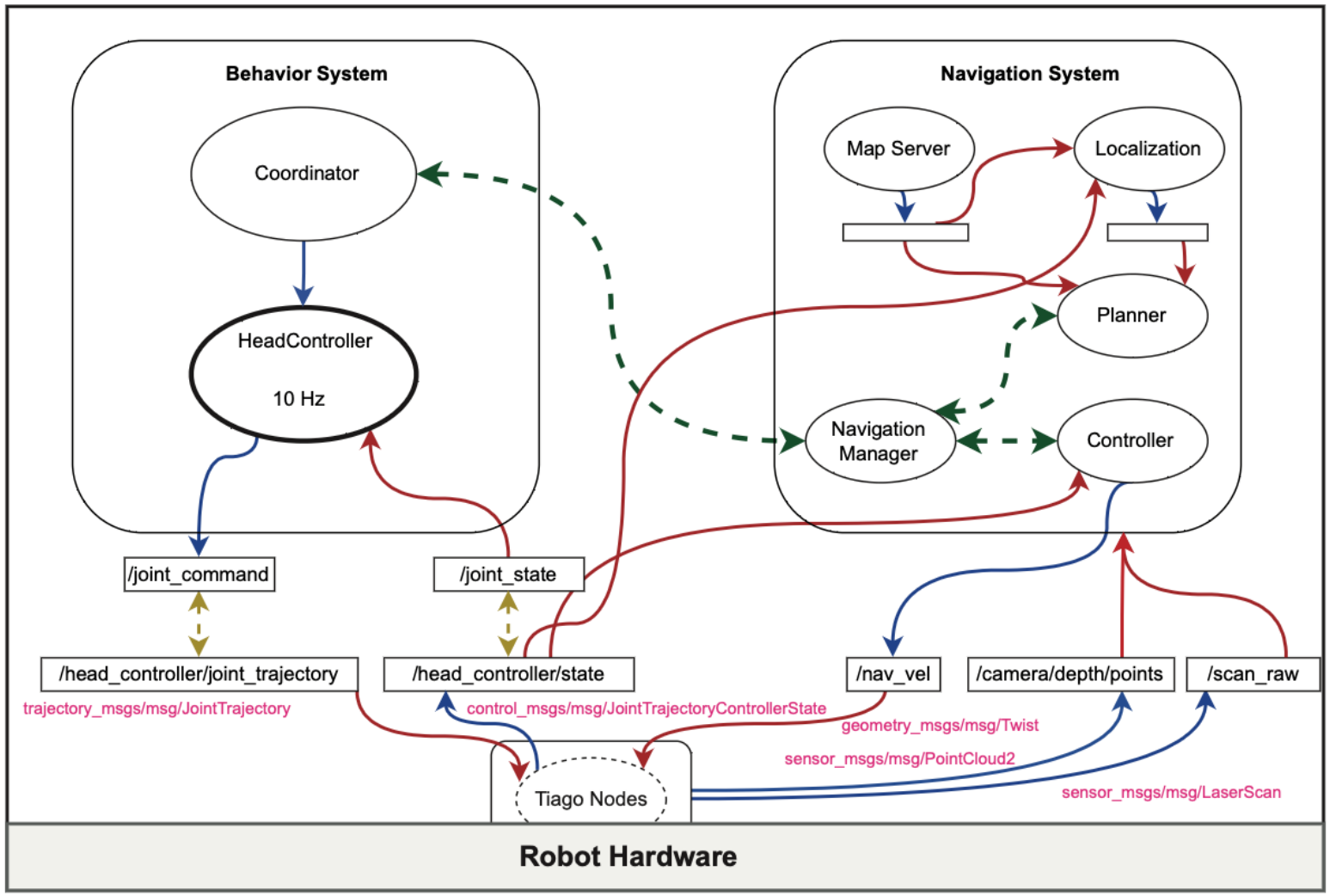
The Computation Graph

Examples

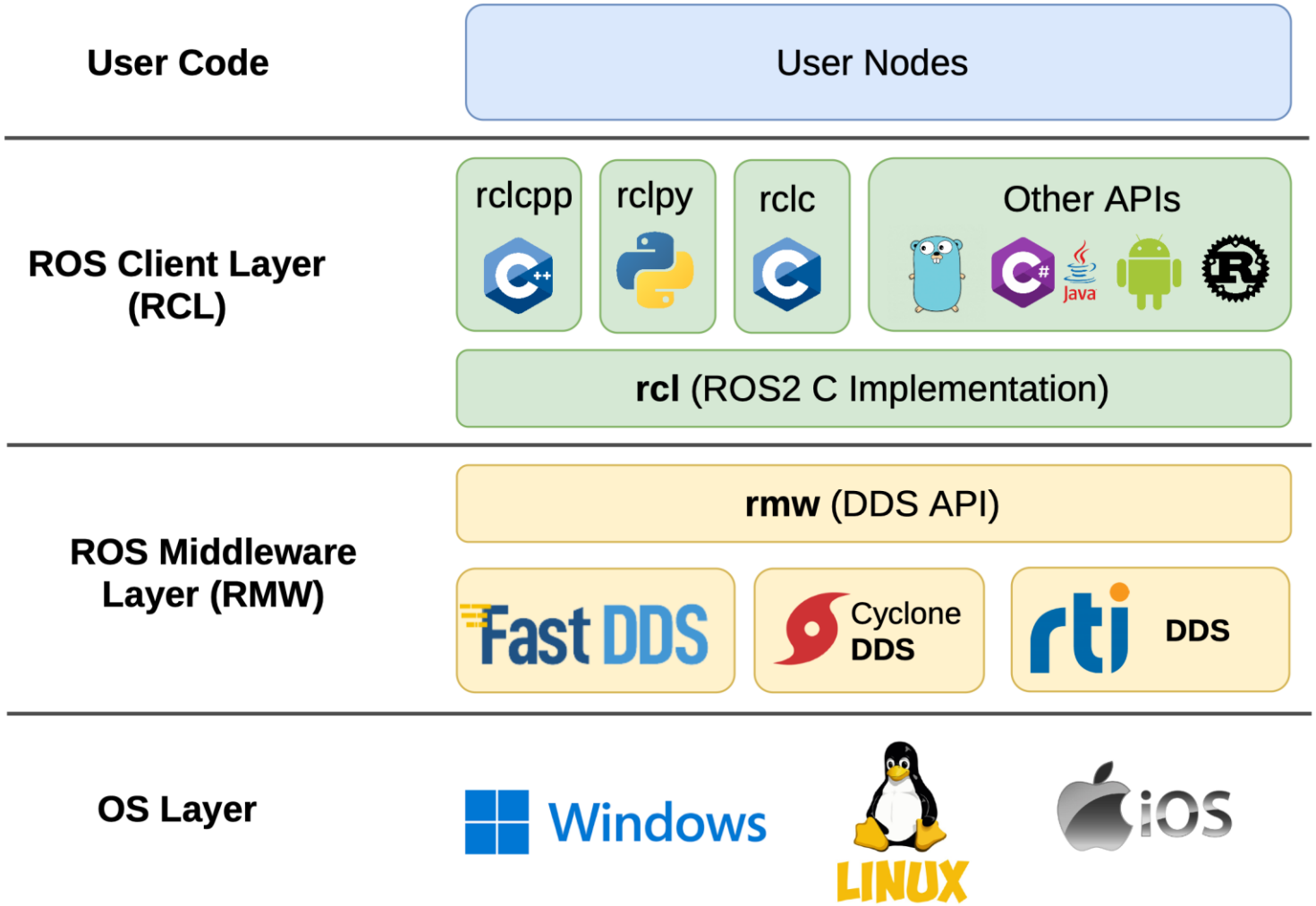


The Computation Graph

Examples



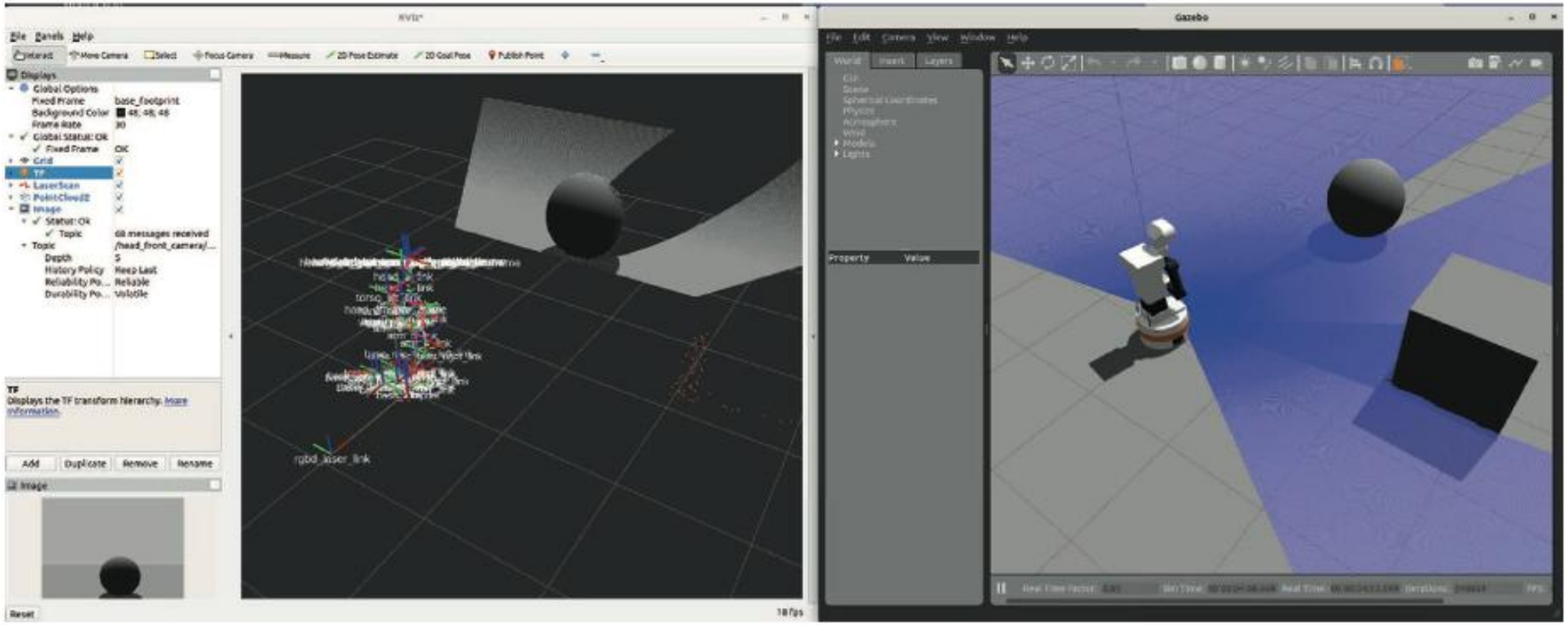
ROS2 Design

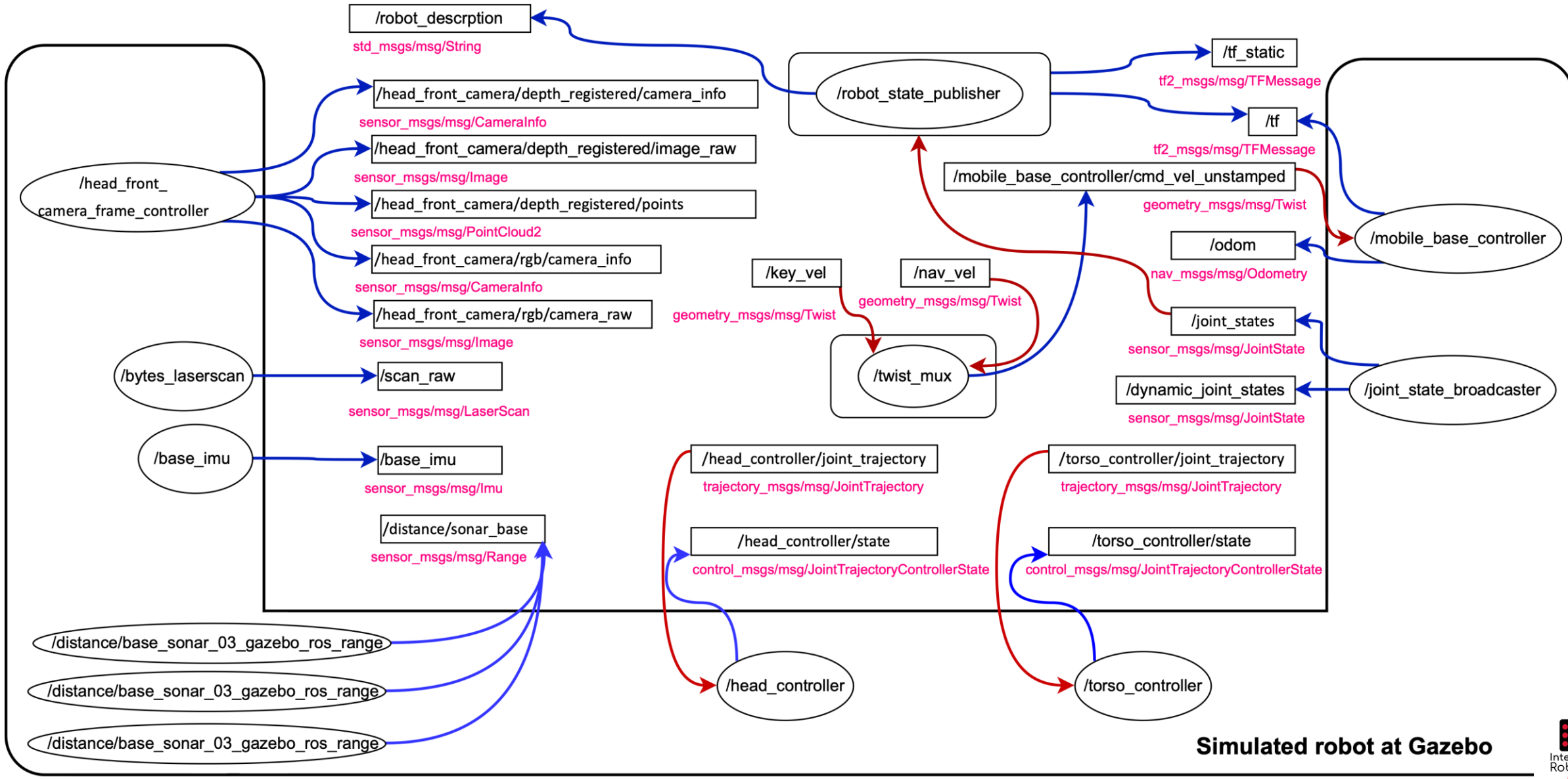

User Code
User Nodes
ROS Client Layer (RCL)
rclcpp
rclpy
rclc
Other APIs
Java
rcl (ROS2 C Implementation)
ROS Middleware Layer (RMW)
rmw (DDS API)
Fast DDS
Cyclone DDS
rti DDS
OS Layer
Windows
LINUX
iOS

Simulated Robot Setup

Rviz2, Gazebo

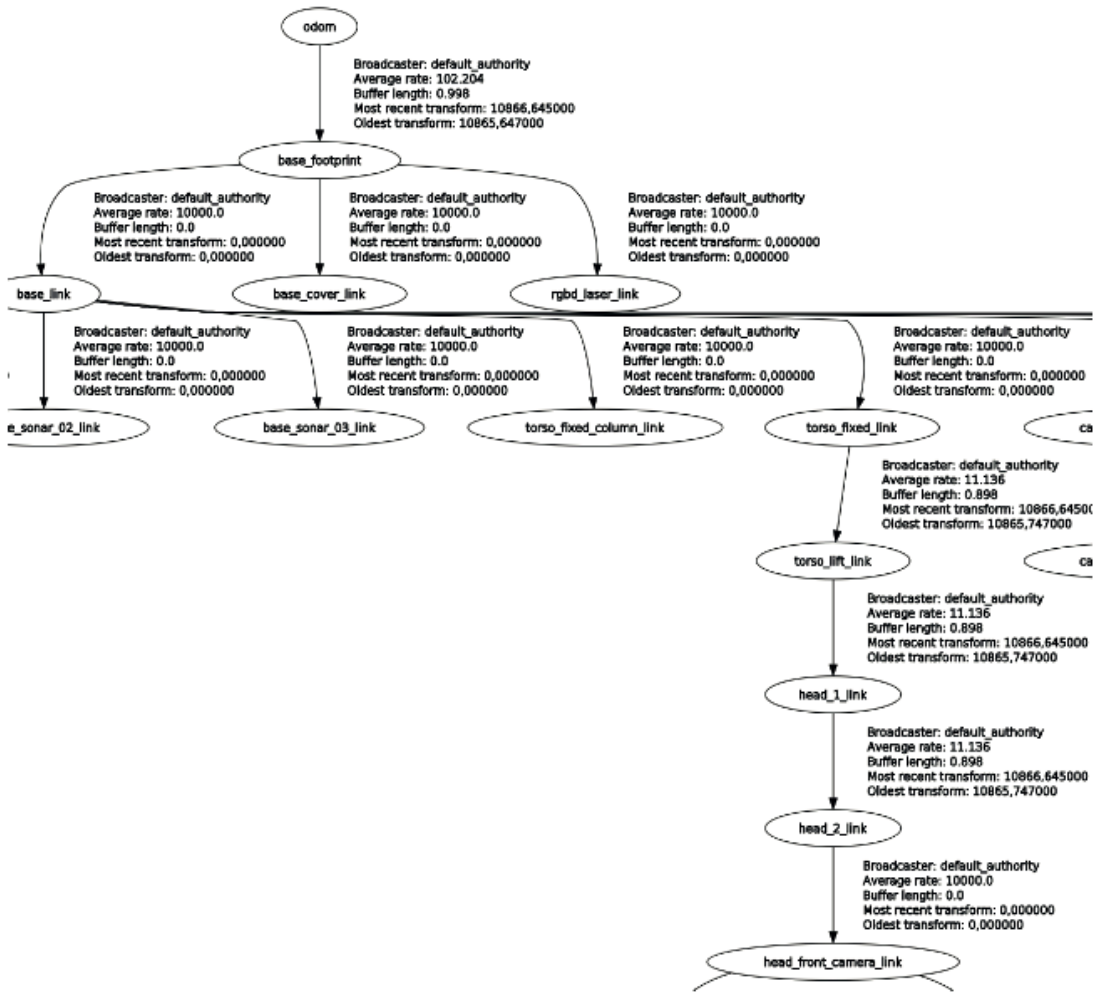
```
$ ros2 run rviz2 rviz2
```

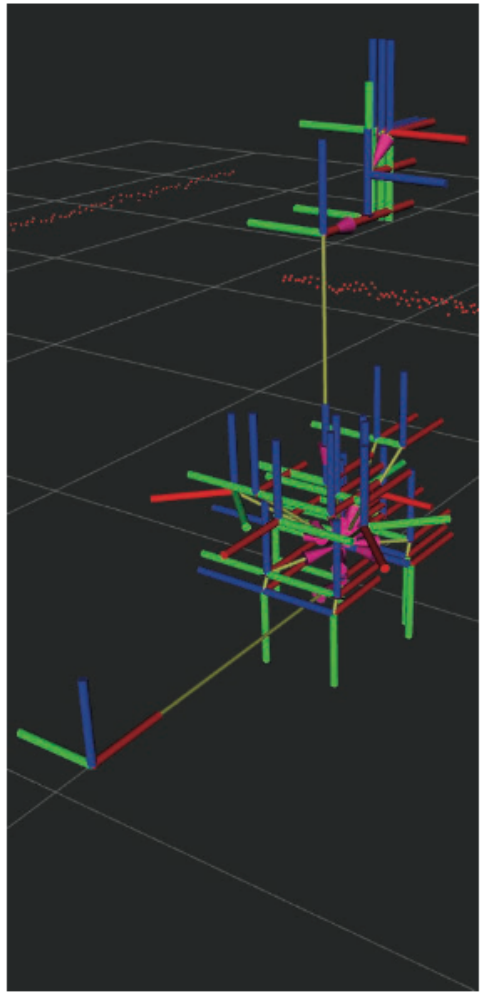



/robot_descrption
std_msgs/msg/String
/robot_state_publisher
/tf_static
tf2_msgs/msg/TFMessage
/tf
tf2_msgs/msg/TFMessage
/head_front_camera/depth_registered/camera_info
sensor_msgs/msg/CameraInfo
/head_front_camera/depth_registered/image_raw
sensor_msgs/msg/Image
/head_front_camera/depth_registered/points
sensor_msgs/msg/PointCloud2
/head_front_camera/rgb/camera_info
sensor_msgs/msg/CameraInfo
/head_front_camera/rgb/camera_raw
sensor_msgs/msg/Image
/head_front_
camera_frame_controller
/mobile_base_controller/cmd_vel_unstamped
geometry_msgs/msg/Twist
/mobile_base_controller
/odom
nav_msgs/msg/Odometry
/key_vel
geometry_msgs/msg/Twist
/nav_vel
geometry_msgs/msg/Twist
/twist_mux
/joint_states
sensor_msgs/msg/JointState
/joint_state_broadcaster
/dynamic_joint_states
sensor_msgs/msg/JointState
/bytes_laserscan
/scan_raw
sensor_msgs/msg/LaserScan
/base_imu
/base_imu
sensor_msgs/msg/Imu
/distance/sonar_base
sensor_msgs/msg/Range
/head_controller/joint_trajectory
trajectory_msgs/msg/JointTrajectory
/torso_controller/joint_trajectory
trajectory_msgs/msg/JointTrajectory
/head_controller/state
control_msgs/msg/JointTrajectoryControllerState
/torso_controller/state
control_msgs/msg/JointTrajectoryControllerState
/distance/base_sonar_03_gazebo_ros_range
/distance/base_sonar_03_gazebo_ros_range
/distance/base_sonar_03_gazebo_ros_range
/head_controller
/torso_controller
Simulated robot at Gazebo

TF Subsystem

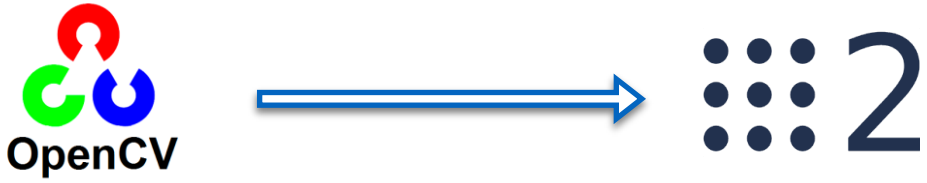
```
$ ros2 run rqt_tf_tree rqt_tf_tree
```

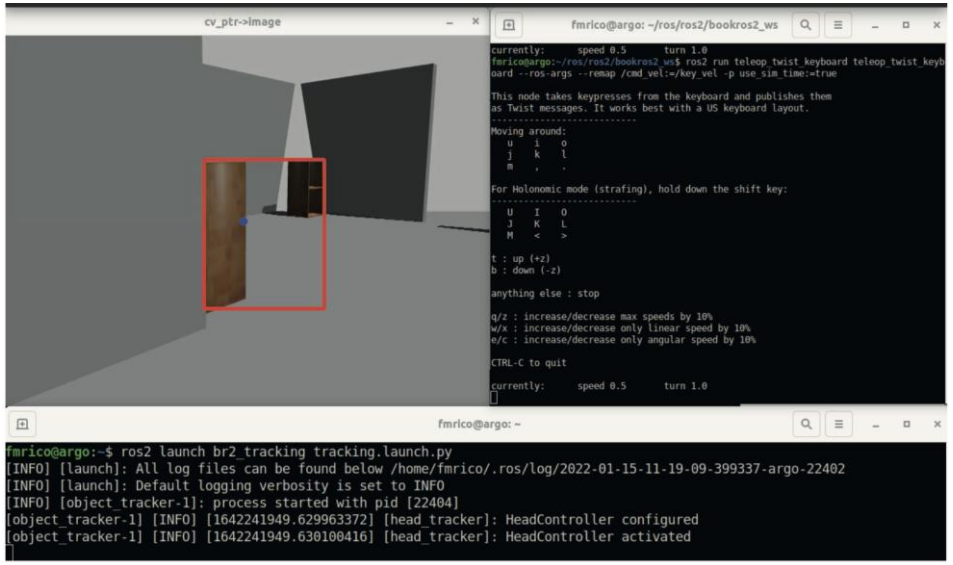


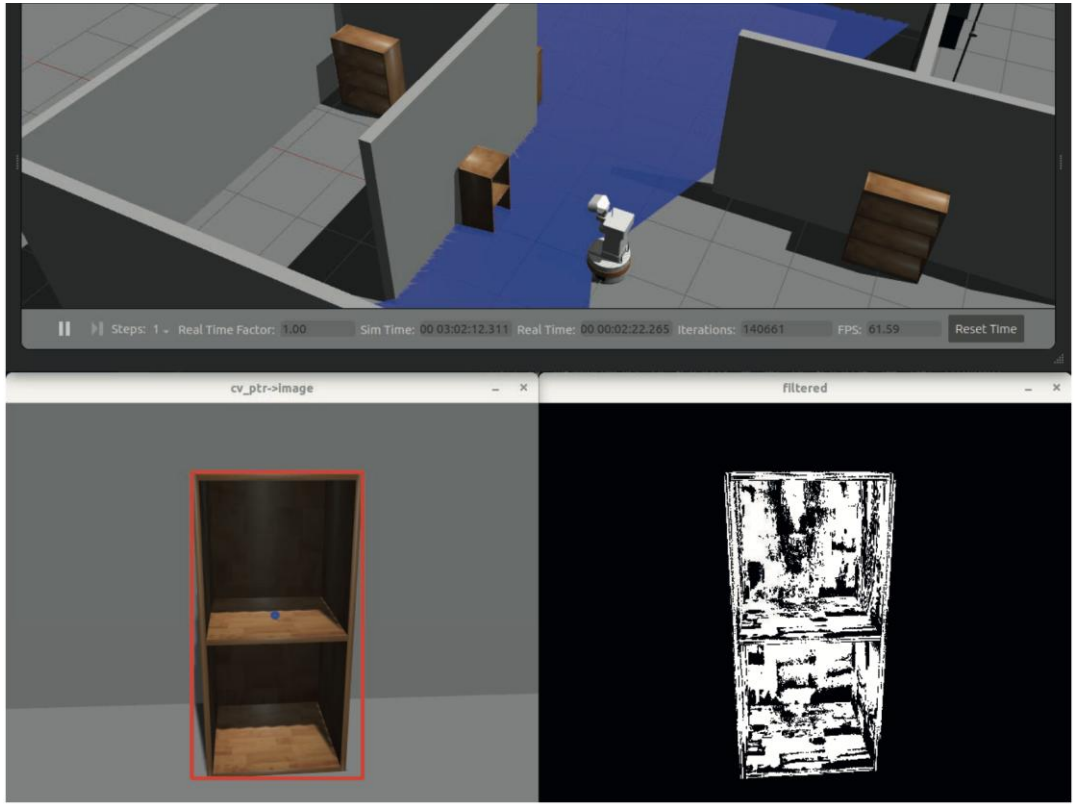


Perception


Gazebo
File Edit Camera View Window Help
Steps: Real Time Factor: 0.98 Sim Time: 00 03:18:08.504 Real Time: 00 00:18:32.479 Iterations: 1096854 FPS: 60.92 Reset Time


OpenCV
2


cv_ptr->image
fmrico@argo: ~/ros/ros2/bookros2_ws
fmrico@argo: ~
fmrico@argo:~$ ros2 launch br2_tracking tracking.launch.py
[INFO] [launch]: All log files can be found below /home/fmrico/.ros/log/2022-01-15-11-19-09-399337-argo-22402
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [object_tracker-1]: process started with pid [22404]
[object_tracker-1] [INFO] [1642241949.629963372] [head_tracker]: HeadController configured
[object_tracker-1] [INFO] [1642241949.630100416] [head_tracker]: HeadController activated


Steps: 1 Real Time Factor: 1.00 Sim Time: 00 03:02:12.311 Real Time: 00 00:02:22.265 Iterations: 140661 FPS: 61.59 Reset Time
cv_ptr->image
filtered

ROS: logging

rosbag: This is a set of tools for recording from and playing back to ROS topics. It can be used to mimic real sensor streams for offline debugging.

http://www.ros.org/wiki/rosbag

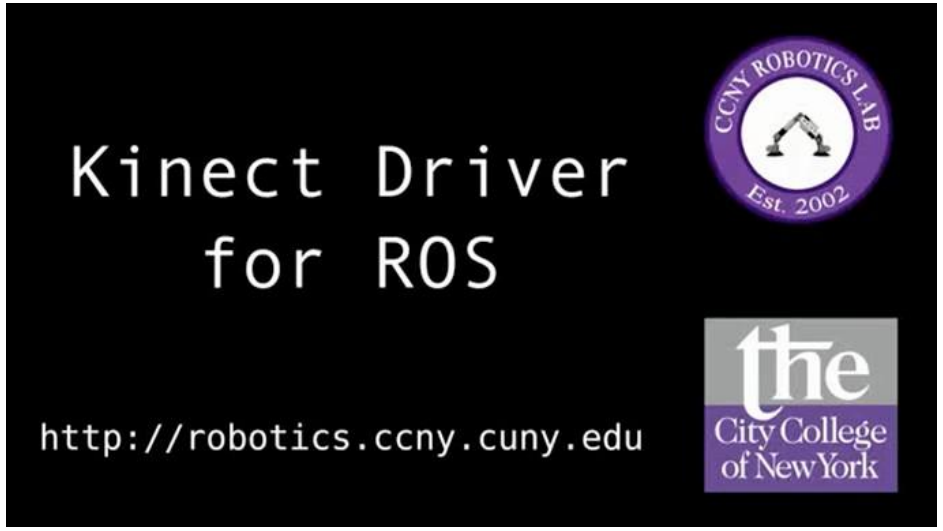
ROS: device drivers

Problem:

Many sensors do not come with standardized interfaces. Often the manufacturer only provides support for a single operating system (e.g. Microsoft Windows).

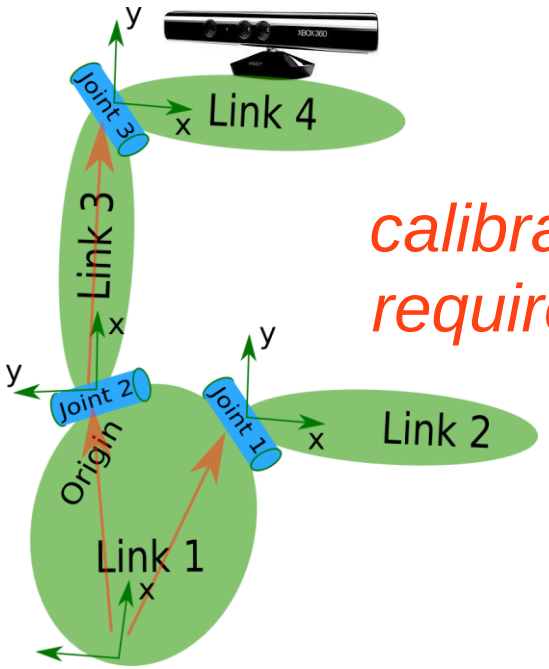
Thus, everybody that wants to use a particular sensor is required to write their own device driver, which is time consuming and tedious.

Instead, a few people did the work and the rest of the world (re-)uses their code and builds on top of it.



ROS: robot descriptions

urdf: This package contains a C++ parser for the **U**nified **R**obot **D**escription **F**ormat (URDF), which is an XML format for representing a robot model.



calibration required !!

http://www.ros.org/wiki/urdf

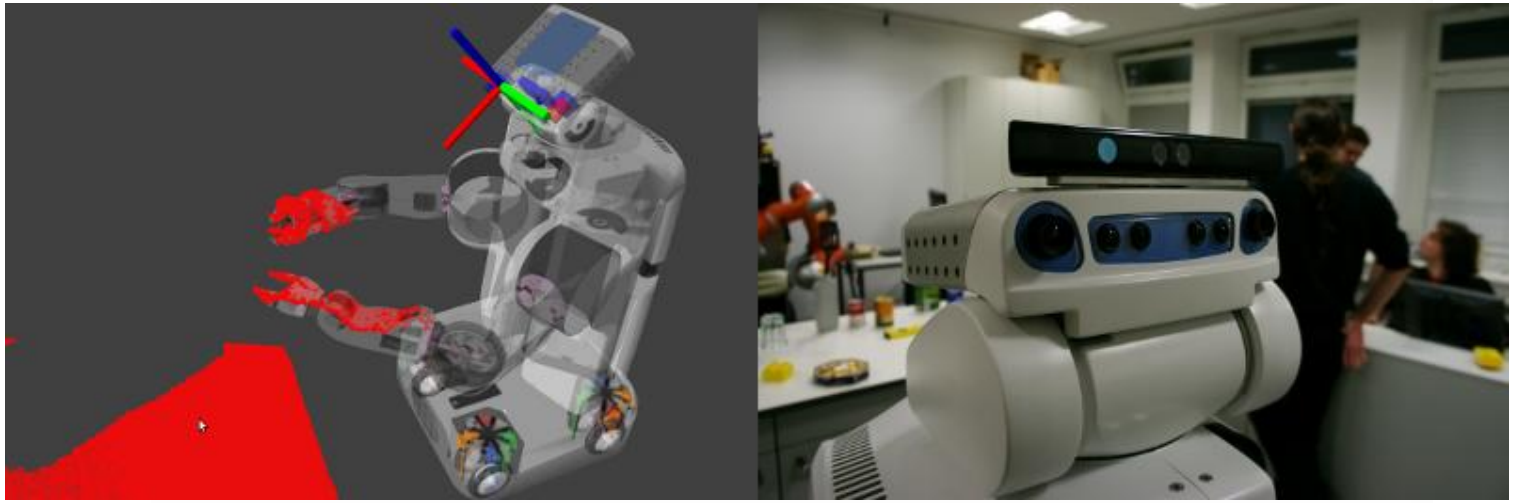
```
<robot name="test_robot">
  <link name="link1" />
                    "
  <link name="link2" />
                    "
  <link name="link3" />
                    "
  <link name="link4" />
                    "
  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="kinect_link"/>
  </joint>
</robot>
```

```
  </joint>
</robot>
```


ROS: calibration

Provides a toolchain running through the robot calibration process. This involves capturing pr2 calibration data, estimating pr2 parameters, and then updating the PR2 URDF.



http://www.ros.org/wiki/pr2_calibration

ROS: visualization

rviz: This is a 3D visualization environment for robots. It allows you to see the world through the eyes of the robot.

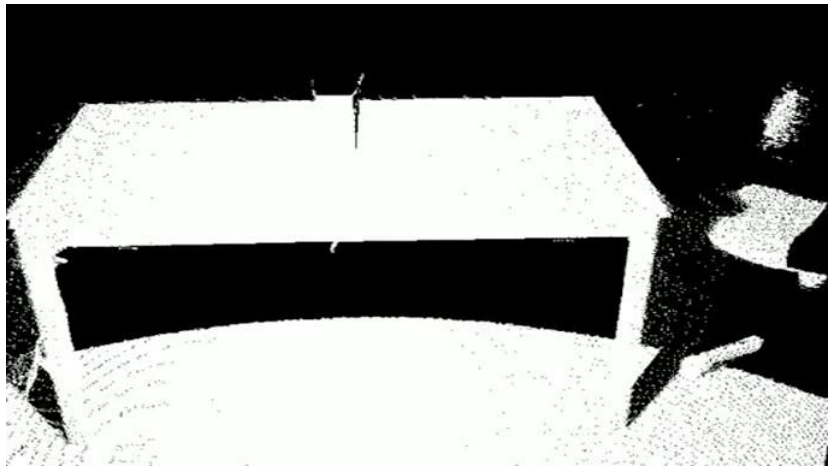
http://www.ros.org/wiki/rviz

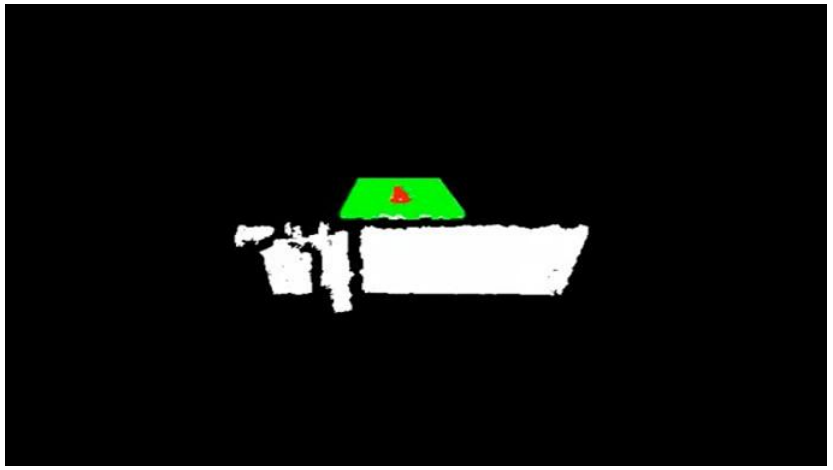
ROS: 2D/3D perception

OpenCV: (**Open** Source **C**omputer **V**ision) is a library of programming functions for real time computer vision. http://opencv.willowgarage.com/wiki/

Check out CS 574 (Prof. Ram Nevatia) !!

PCL - **P**oint **C**loud **L**ibrary: a comprehensive open source library for **n-D Point Clouds** and **3D geometry processing**. The library contains numerous state-of-the art algorithms for: filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation, etc.

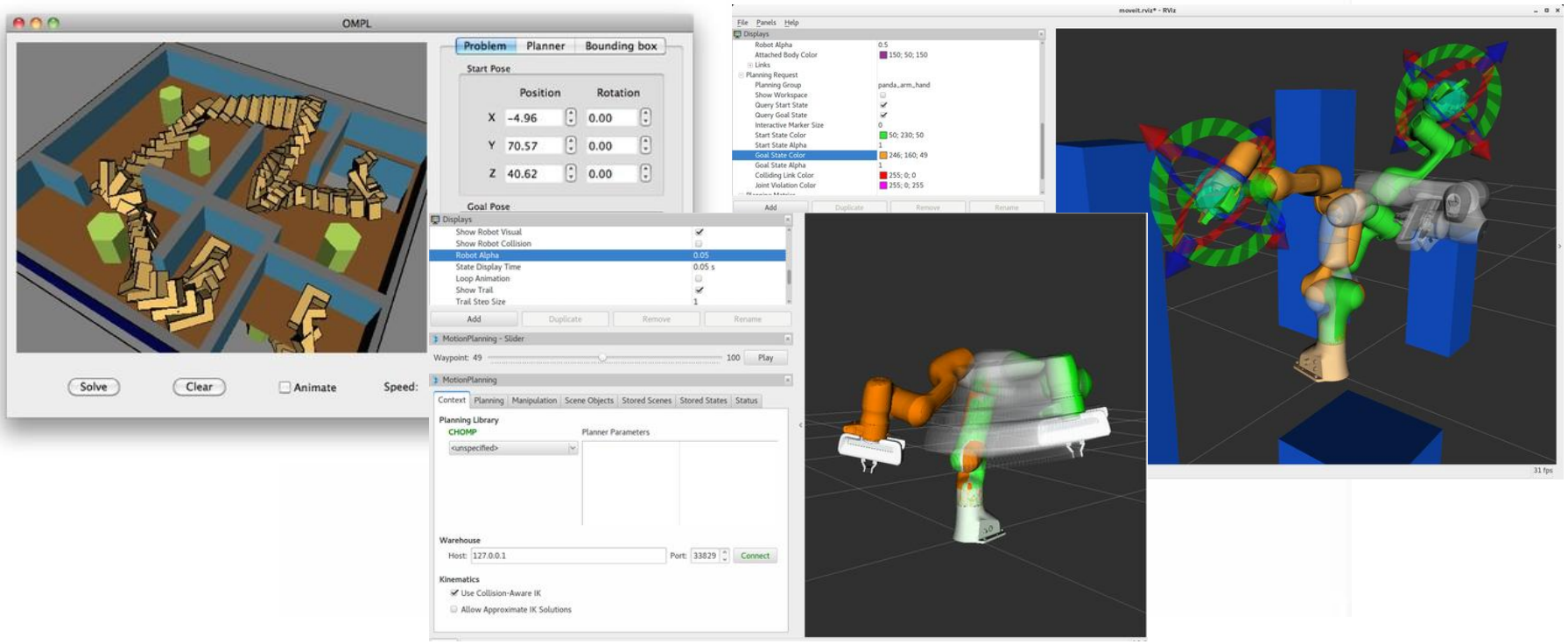




http://www.ros.org/wiki/pcl

ROS: planning

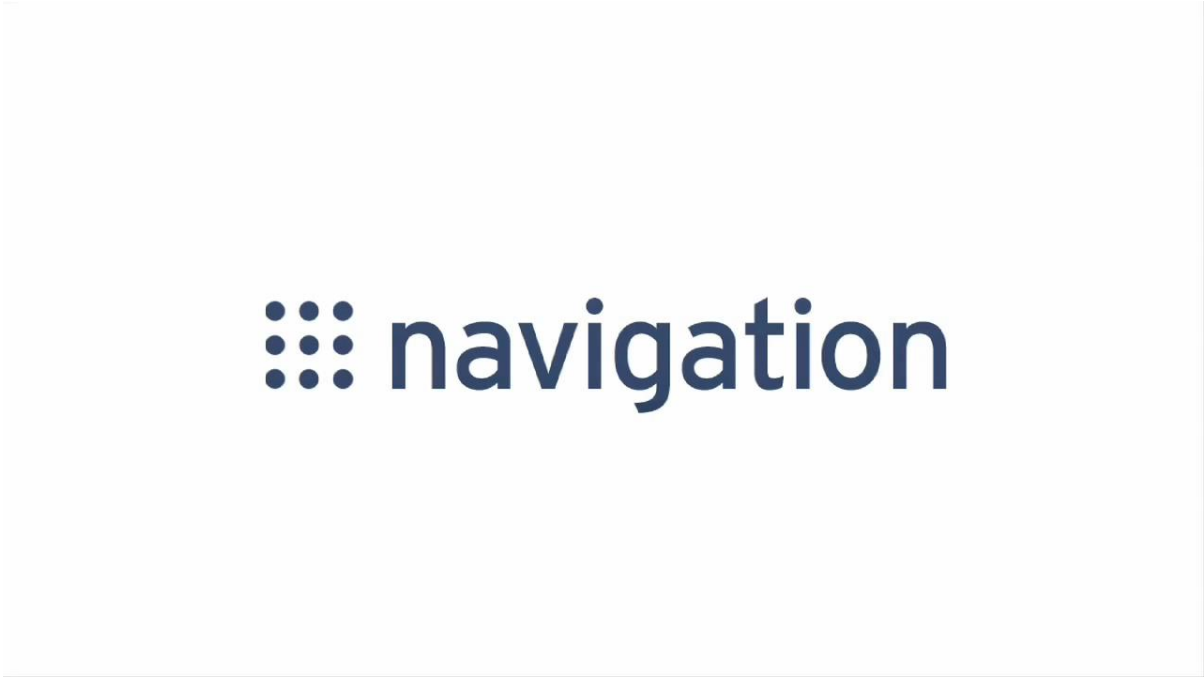
The **motion_planners** stack contains different motion planners including probabilistic motion planners, search-based planners, and motion planner based on trajectory optimization.



http://www.ros.org/wiki/motion_planners

ROS: navigation

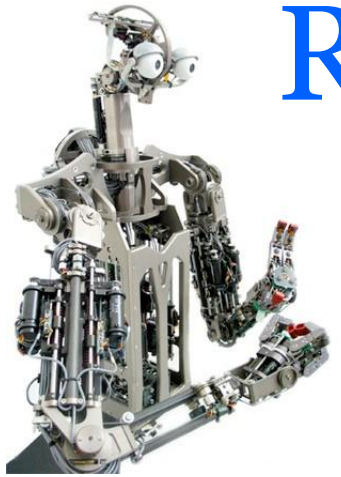
navigation: A 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base.



http://www.ros.org/wiki/navigation

Example application

Overview



ROS

navigation

task executive

visualization

simulation

perception

control

planning

data logging

message passing

device drivers

real-time capabilities

OS

web browser

email client

window manager

memory management

process management

scheduler

device drivers

file system