Lab 03
ROS Simulation

Wan Fang

Visiting Scholar

SUSTech Institute of Robotics

sophie.fwan@hotmail.com

Agenda

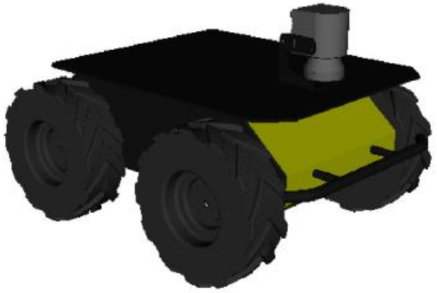
Week 03, Friday, Mar 8

- ROS Simulation
  - Robot/Scene Description: URDF
  - ROS simulation: Gazebo
  - ROS Control
  - Motion Planning: MoveIt
- Home work: Franka

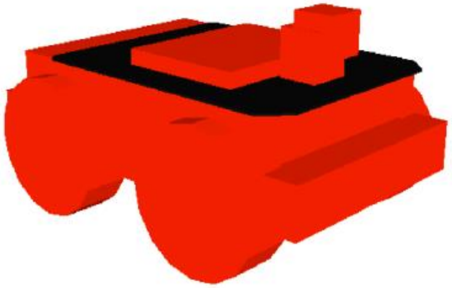
URDF

Unified Robot Description Format

- ***URDF*** Defines an XML format for representing a robot model
  - Kinematic and dynamic description
  - Visual representation
  - Collision model



Mesh for visuals



Primitives for collision

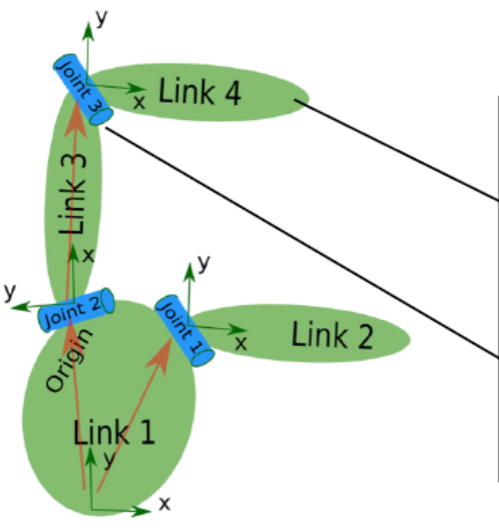
- Define working scene of the robot
- URDF generation can be scripted with XACRO

More info
http://wiki.ros.org/urdf， http://wiki.ros.org/xacro

URDF

Link

- ***Link*** description contains
  - name
  - visual: size, color, shape, geometry primitives, meshes, material
  - inertial matrix, collision properties.
- Every link is a coordinate/frame



robot.urdf

```
<robot name="robot">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

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

```
<link name="link_name">
  <visual>
    <geometry>
      <mesh filename="mesh.dae"/>
    </geometry>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
  <inertial>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" …/>
  </inertial>
</link>
```

URDF

Joint

- ***Joint*** describe the relationship between two links, properties including
  - Name, type, parent, child, origin (transform from the parent link to the child link), axis
  - Limit: lower and upper rotation/translation limits, max velocity, max effort
  - Kinematics: one joint follows another joint
  - Dynamics: friction, damping

```
<joint name="joint_name" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort="1000.0" upper="0.548" … />
  <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
  <parent link="parent_link_name"/>
  <child link="child_link_name"/>
</joint>
```

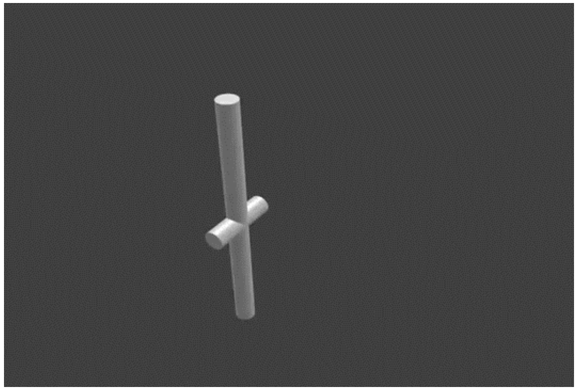
AncoraSIR.com

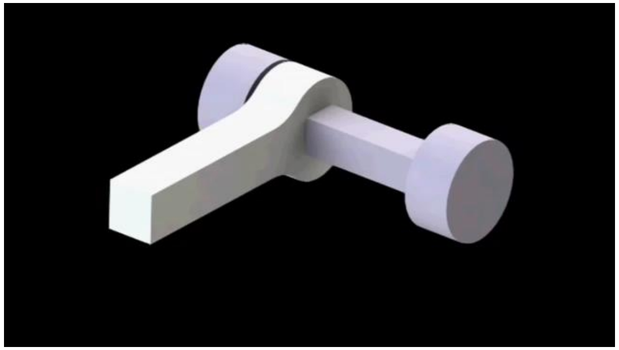
SUSTech
Southern University of Science and Technology

URDF

Joint

Joint Types	Description
continuous	A continuous hinge joint that rotates around the axis and has no upper and lower limits.
revolute	A hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits.
prismatic	A sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits.
planar	This joint allows motion in a plane perpendicular to the axis.
floating	This joint allows motion for all 6 degrees of freedom.
fixed	This is not really a joint because it cannot move. All degrees of freedom are locked.





URDF

Joint

- ***<u>Transmission</u>*** describe the relationship between an actuator and a joint. This allows one to model concepts such as gear ratios and parallel linkages.
- Used together with ***<u>ros_control</u>***.

```
<transmission name="simple_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="foo_joint">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="foo_motor">
    <mechanicalReduction>50</mechanicalReduction>
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </actuator>
</transmission>
```

AncoraSIR.com

More info: http://wiki.ros.org/urdf/XML/Transmission

SUSTech Southern University of Science and Technology

URDF

Building Franka Model

```
<xacro:macro name="cylinder_inertial" params="radius length mass *origin">
<xacro:macro name="panda_arm" params="arm_id:='panda' description_pkg:='franka_description' connected
  <xacro:unless value="${not connected_to}">
  <link name="${arm_id}_link0">
  <link name="${arm_id}_link1">
  <joint name="${arm_id}_joint1" type="revolute">
  <link name="${arm_id}_link2">
  <joint name="${arm_id}_joint2" type="revolute">
  <link name="${arm_id}_link3">
  <joint name="${arm_id}_joint3" type="revolute">
  <link name="${arm_id}_link4">
  <joint name="${arm_id}_joint4" type="revolute">
```

```
in_ws/src/BionicDL-CobotLearning-Project1/franka_description/robots/panda_arm.xacro   18:56   LF   UTF-8   XML   master   Fetch   4 files
```

A list of robots described by URDF files can be found here: http://wiki.ros.org/urdf/Examples

URDF

panda_arm.xacro — ~/catkin_ws/src/BionicDL-CobotLearning-Project1/franka_description/robots — Atom

franka.launch | panda_arm.xacro

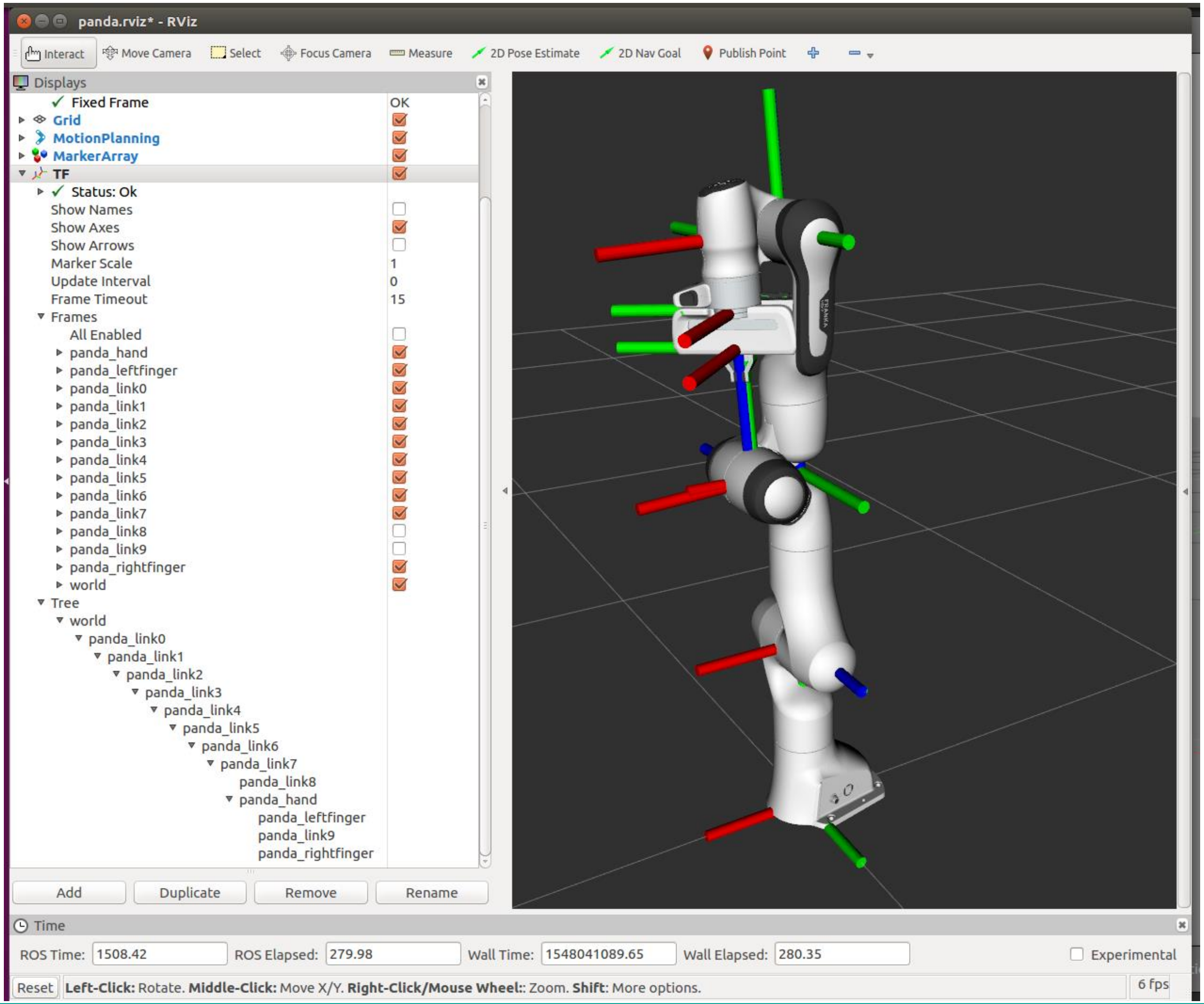
```
<xacro:macro name="panda_arm" params="arm_id:='panda' description_pkg:='franka_description' connected

  <xacro:unless value="${not connected_to}">

  <link name="${arm_id}_link0">
    <visual>
      <geometry>
        <mesh filename="package://${description_pkg}/meshes/visual/link0.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <mesh filename="package://${description_pkg}/meshes/collision/link0.stl"/>
      </geometry>
    </collision>
    <xacro:cylinder_inertial radius="0.06" length="${link0_length}" mass="${link0_mass}">
      <origin xyz="0.0 0.0 0.0" rpy="0 0 0" />
    </xacro:cylinder_inertial>
  </link>

  <link name="${arm_id}_link1">
    <visual>
      <geometry>
        <mesh filename="package://${description_pkg}/meshes/visual/link1.dae"/>
      </geometry>
    </visual>
    <collision>
      <geometry>
        <mesh filename="package://${description_pkg}/meshes/collision/link1.stl"/>
      </geometry>
    </collision>
    <xacro:cylinder_inertial radius="0.06" length="${link1_length}" mass="${link1_mass}">
      <origin xyz="0.0 0.0 0.0" rpy="0 0 0" />
    </xacro:cylinder_inertial>
  </link>

  <joint name="${arm_id}_joint1" type="revolute">
    <!-- <safety_controller k_position="100.0" k_velocity="40.0" soft_lower_limit="-2.8973" soft_uppe
    <origin rpy="0 0 0" xyz="0 0 0.333"/>
    <parent link="${arm_id}_link0"/>
    <child link="${arm_id}_link1"/>
    <axis xyz="0 0 1"/>
    <limit effort="87" lower="-2.8973" upper="2.8973" velocity="2.1750"/>
  </joint>
```

AncoraSIR.com

SUSTech
Southern University of Science and Technology

URDF

Building Franka Model



AncoraSIR.com

ROS simulation

Gazebo

- ***Gazebo*** is a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces.
  - Useful if you don't have a real robot or camera.
  - Automatically installed with ROS desktop-full.
- What do you need for gazebo simulation?
  - ***World Files***: contains all the elements in a simulation, including robots, lights, sensors, and static objects.
  - ***Model Files***: models of the objects.
  - ***Environment Variables***: set environment variables to locate files, and set up communications between the server and clients.

AncoraSIR.com

SUSTech Southern University of Science and Technology

ROS simulation

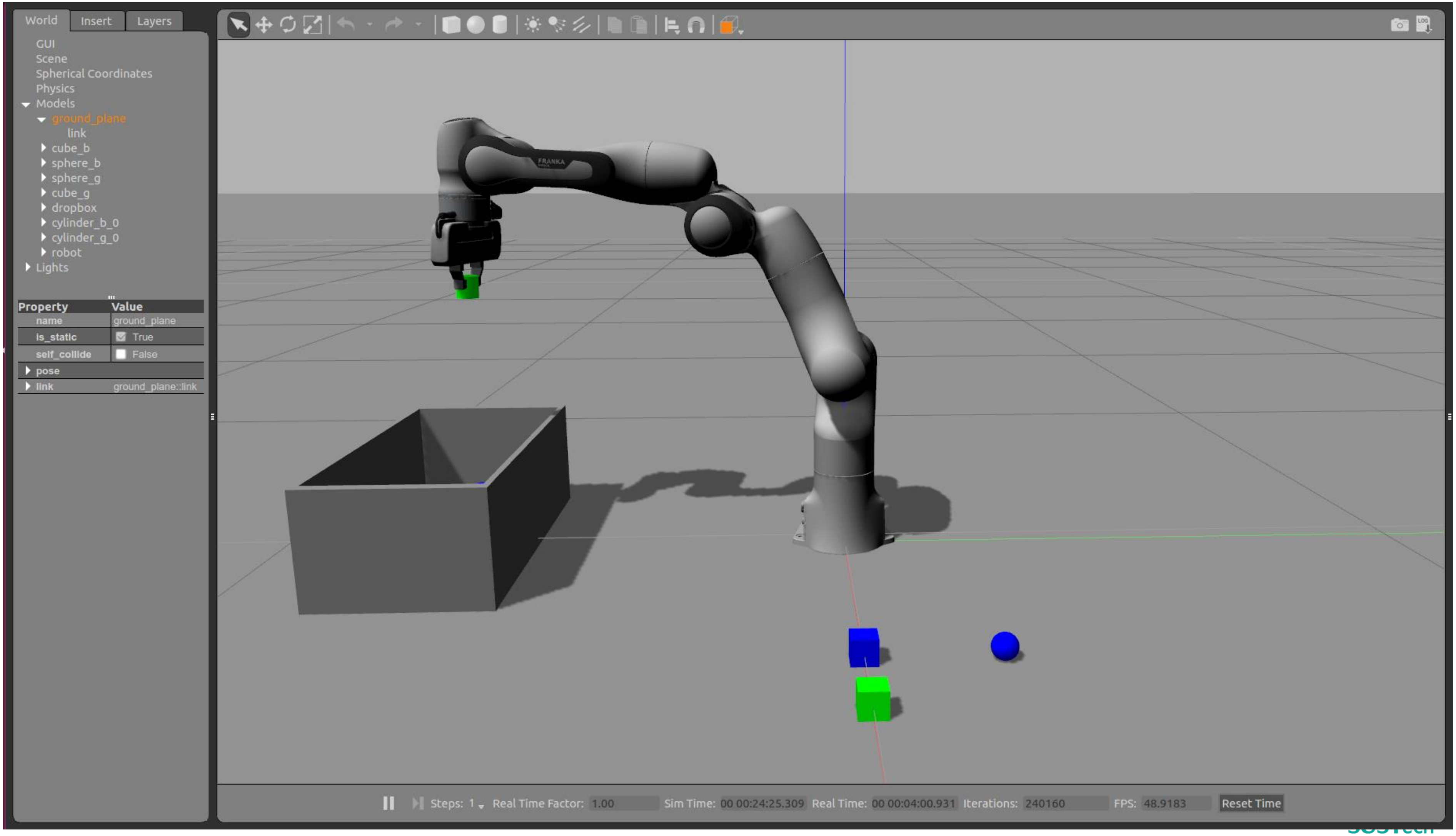
World Files

- ***Simulation Description Format*** (SDF) defines an XML format to describe
  - Environments (lighting, gravity etc.)
  - Objects (static and dynamic)
  - Sensors
  - Robots
- SDF is the standard format for Gazebo
- ***Gazebo*** converts a URDF to SDF automatically

AncoraSIR.com

SUSTech
Southern University of Science and Technology

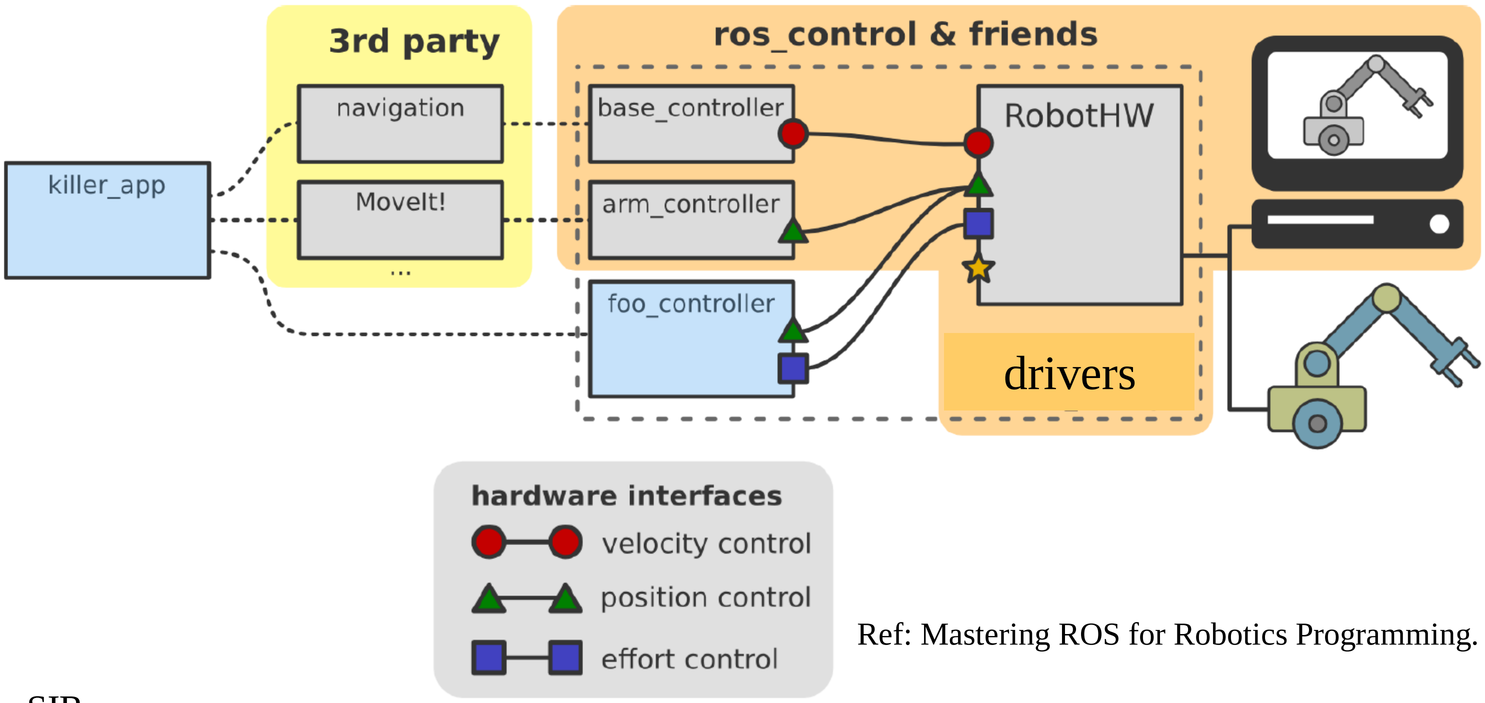
ROS simulation: Gazebo



AncoraSIR.com

ROS Control

- ROS_control is a set of packages that connects application softwares to robotic hardware.
- Include controller interfaces, controller managers, transmissions and hardware_interfaces.
- Lower entry barrier, reuse of control code, Real-time ready implementation

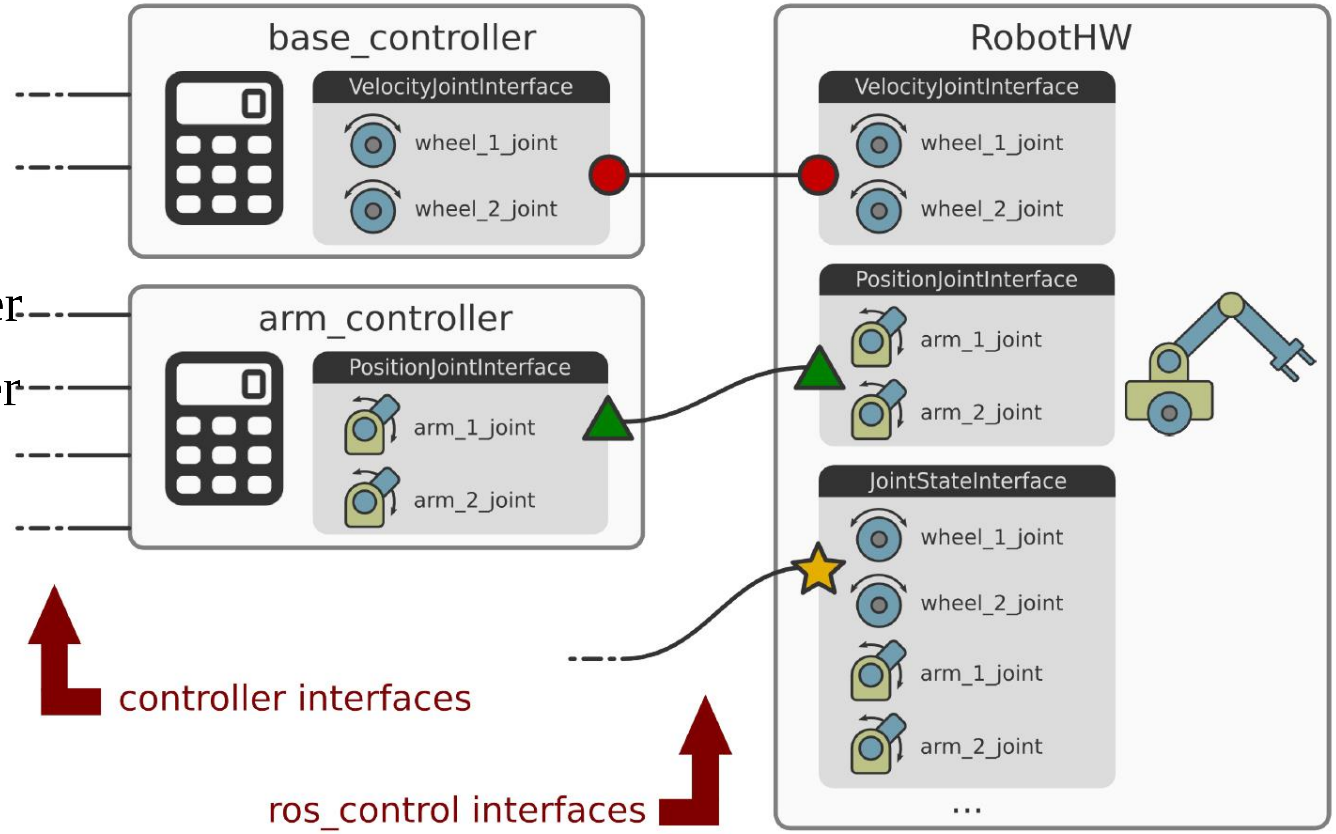


Ref: Mastering ROS for Robotics Programming. Chapter 3.

AncoraSIR.com

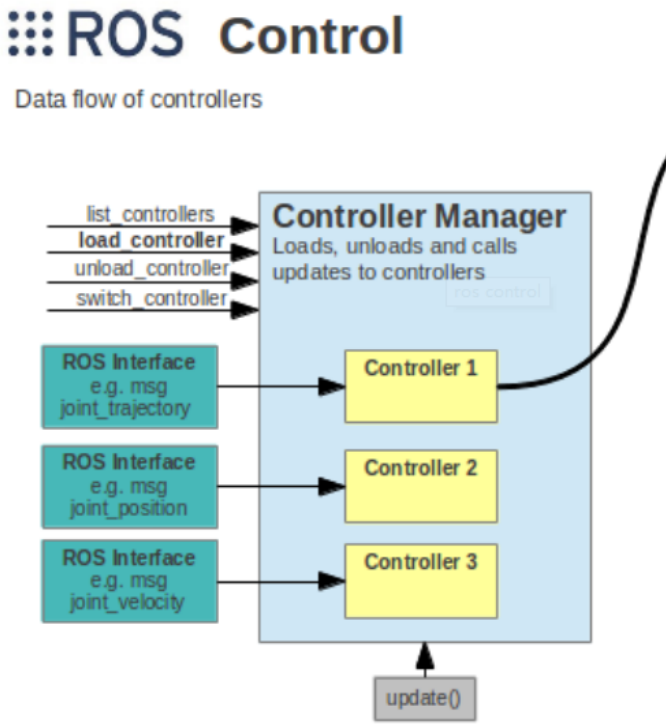
ROS Control

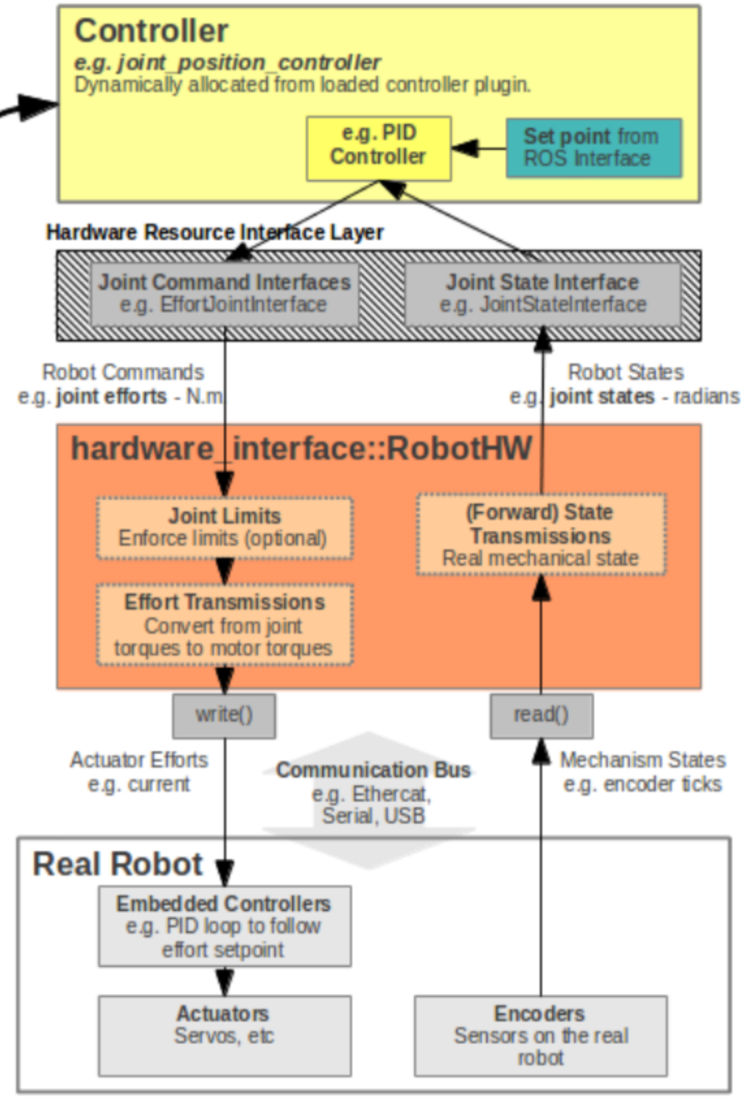
- Controllers:
- joint_state_controller
- joint_effort_controller
- joint_position_controller
- joint_velocity_controller



AncoraSIR.com

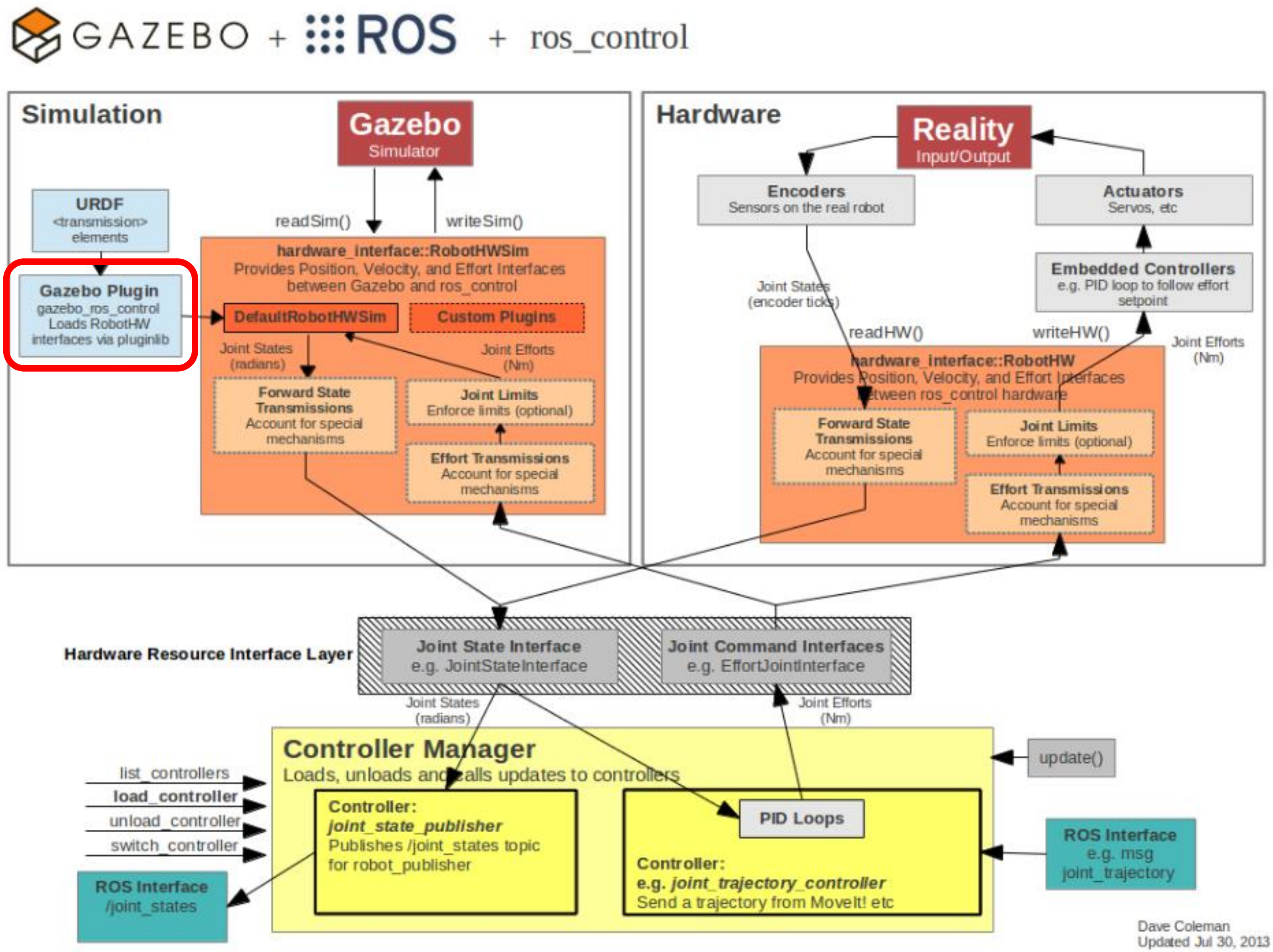
ROS Control





1. Controller Manager: provides the infrastructure to load, unload, start, and stop controllers
2. Controller: request hardware resources, provide PID control, give joint command
3. Hardware Resource Interface: provide hardware sources
4. RobotHW: Robot hardware abstraction(actuators, joints, sensors), talk to HW, provide resources (r. joint state, r.w. position/velocity/effort joint), handles resource conflict
5. Real Robot:

ROS Control



GAZEBO + ROS + ros_control
Simulation
Gazebo
Simulator
URDF
<transmission>
elements
Gazebo Plugin
gazebo_ros_control
Loads RobotHW
interfaces via pluginlib
readSim()
writeSim()
hardware_interface::RobotHWSim
Provides Position, Velocity, and Effort Interfaces between Gazebo and ros_control
DefaultRobotHWSim
Custom Plugins
Joint States
(radians)
Joint Efforts
(Nm)
Forward State Transmissions
Account for special mechanisms
Joint Limits
Enforce limits (optional)
Effort Transmissions
Account for special mechanisms
Hardware
Reality
Input/Output
Encoders
Sensors on the real robot
Actuators
Servos, etc
Embedded Controllers
e.g. PID loop to follow effort setpoint
Joint States
(encoder ticks)
readHW()
writeHW()
Joint Efforts
(Nm)
hardware_interface::RobotHW
Provides Position, Velocity, and Effort Interfaces between ros_control hardware
Forward State Transmissions
Account for special mechanisms
Joint Limits
Enforce limits (optional)
Effort Transmissions
Account for special mechanisms
Hardware Resource Interface Layer
Joint State Interface
e.g. JointStateInterface
Joint Command Interfaces
e.g. EffortJointInterface
Joint States
(radians)
Joint Efforts
(Nm)
Controller Manager
Loads, unloads and calls updates to controllers
update()
list_controllers
load_controller
unload_controller
switch_controller
Controller:
joint_state_publisher
Publishes /joint_states topic for robot_publisher
PID Loops
Controller:
e.g. joint_trajectory_controller
Send a trajectory from MoveIt! etc
ROS Interface
/joint_states
ROS Interface
e.g. msg
joint_trajectory
Dave Coleman
Updated Jul 30, 2013

ROS Control

gazebo-ros-control

- Simulate a moveable robot arm in Gazebo
- needs addition configuration for each joint

```yaml
arm_controller:
  type: "position_controllers/JointTrajectoryController"
  joints:
     - panda_joint1
     - panda_joint2
     - panda_joint3
     - panda_joint4
     - panda_joint5
     - panda_joint6
     - panda_joint7
  gains:
      panda_joint1: {p: 100, i: 0.01, d: 1}
      panda_joint2: {p: 100, i: 0.01, d: 1}
      panda_joint3: {p: 100, i: 0.01, d: 1}
      panda_joint4: {p: 100, i: 0.01, d: 1}
      panda_joint5: {p: 100, i: 0.01, d: 1}
      panda_joint6: {p: 100, i: 0.01, d: 1}
      panda_joint7: {p: 100, i: 0.01, d: 1}
  constraints:
      goal_time: 10.0
  stop_trajectory_duration: 0.0
  state_publish_rate:  50
  action_monitor_rate: 30
```

```yaml
gripper_controller:
  type: "effort_controllers/JointTrajectoryController"
  joints:
     - panda_finger_joint1
     - panda_finger_joint2
  gains:
    panda_finger_joint1: {p: 100, i: 1, d: 10, i_clamp: 1.0}
    panda_finger_joint2: {p: 100, i: 1, d: 10, i_clamp: 1.0}
  constraints:
    goal_time: 4.0
    panda_finger_joint1:
      goal: 0.03
    panda_finger_joint2:
      goal: 0.03
```

SUSTech
Southern University of Science and Technology

ROS Control

gazebo-ros-control

- Add gazebo-ros-control package in the URDF file.
- Add transmission in the URDF file

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
  </plugin>
</gazebo>

<transmission name="${prefix}_joint1_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="${prefix}_joint1">
    <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="${prefix}_joint1_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

Anc

Motion Planning

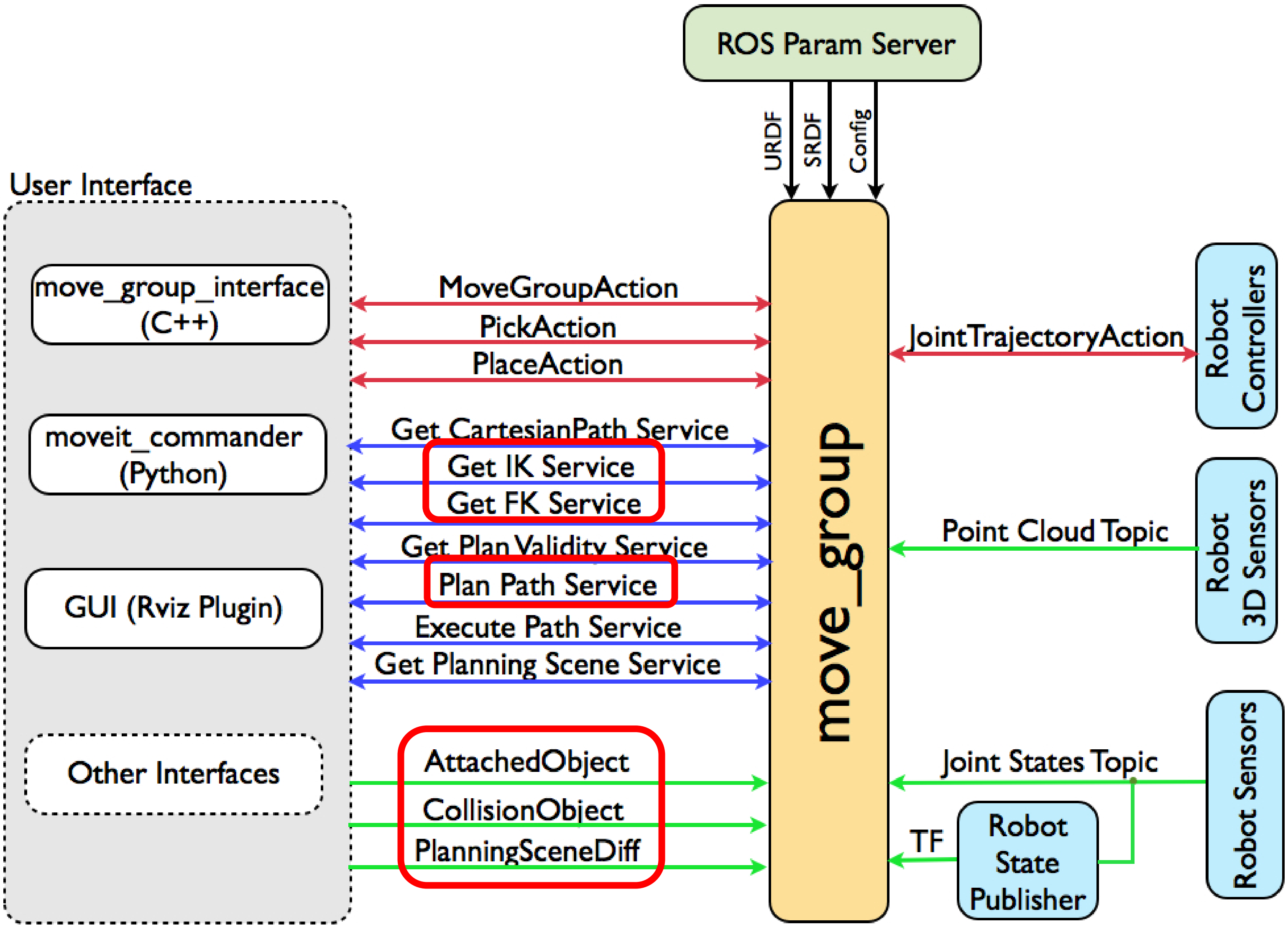
MoveIt

- MoveIt! is a set of packages and tools for doing mobile manipulation in ROS.
- MoveIt! contains state of the art software for motion planning, manipulation, 3D perception, kinematics, collision checking, control, and navigation.
- Installation: $sudo apt-get install ros-kinetic-moveit

Ref: Mastering ROS for Robotics Programming. Chapter 4.
http://moveit.ros.org/documentation/concepts/

Motion Planning

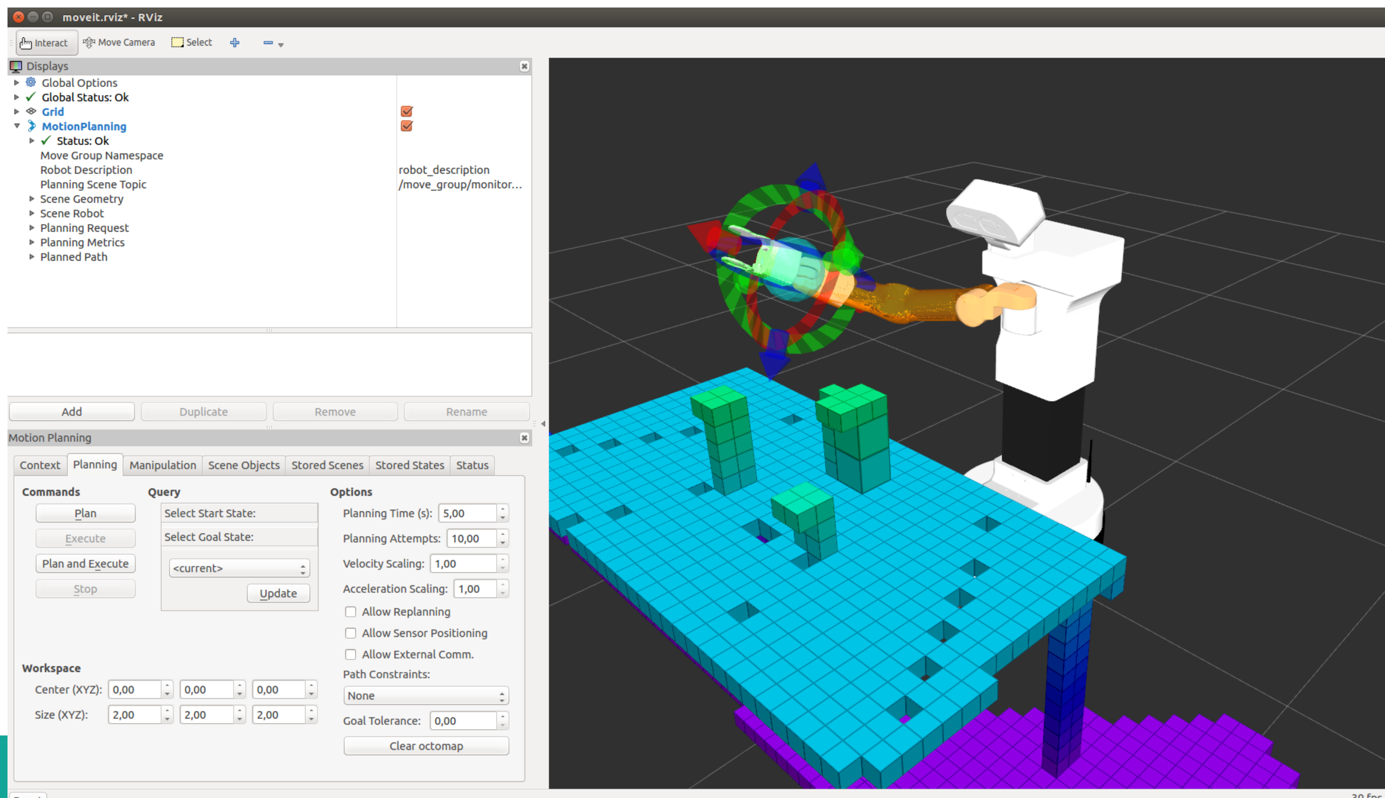
MoveIt Architecture



Motion Planning

Planning Scene

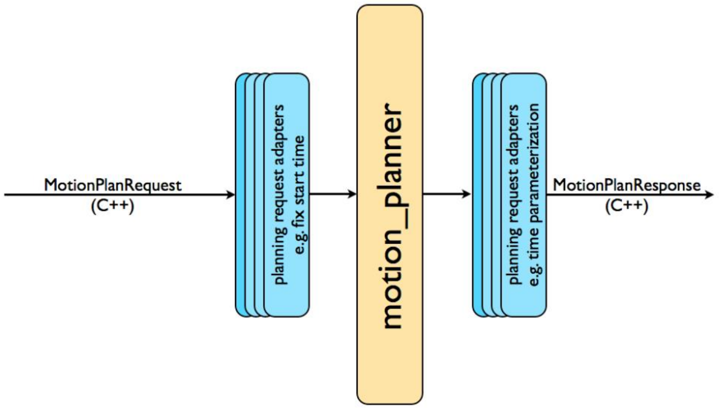
- Represent the world around the robot and also store the state of the robot itself using Octomap
- Read the joint_states topic from the robot, and the sensor information and world geometry from the world geometry monitor

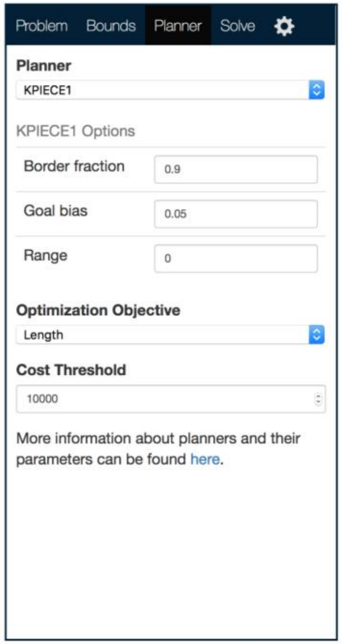


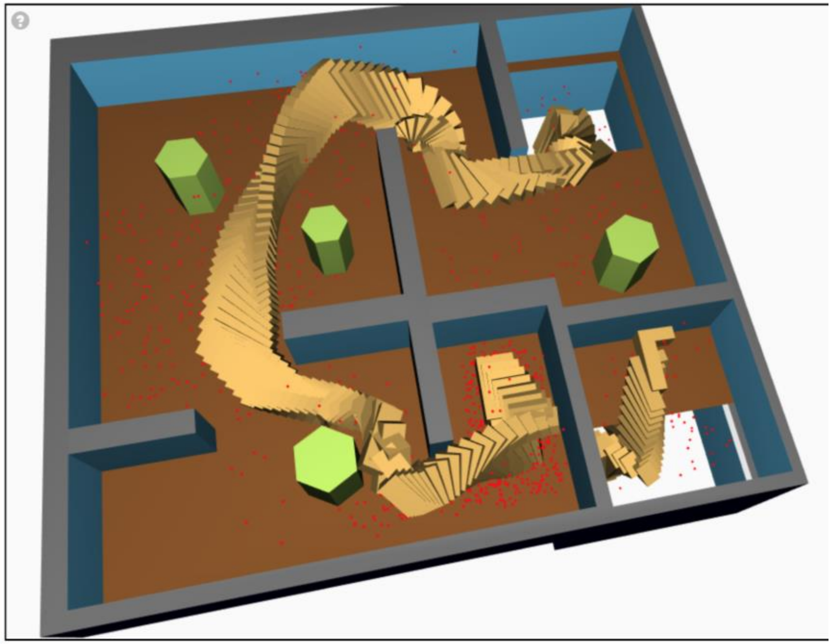
Motion Planning

Motion Planners

- MoveIt! works with motion planners through a plugin interface. This allows MoveIt! to communicate with and use different motion planners from multiple libraries.
- Default library is OMPL(Open Motion Planning Library)







AncoraSIR.com

Motion Planning

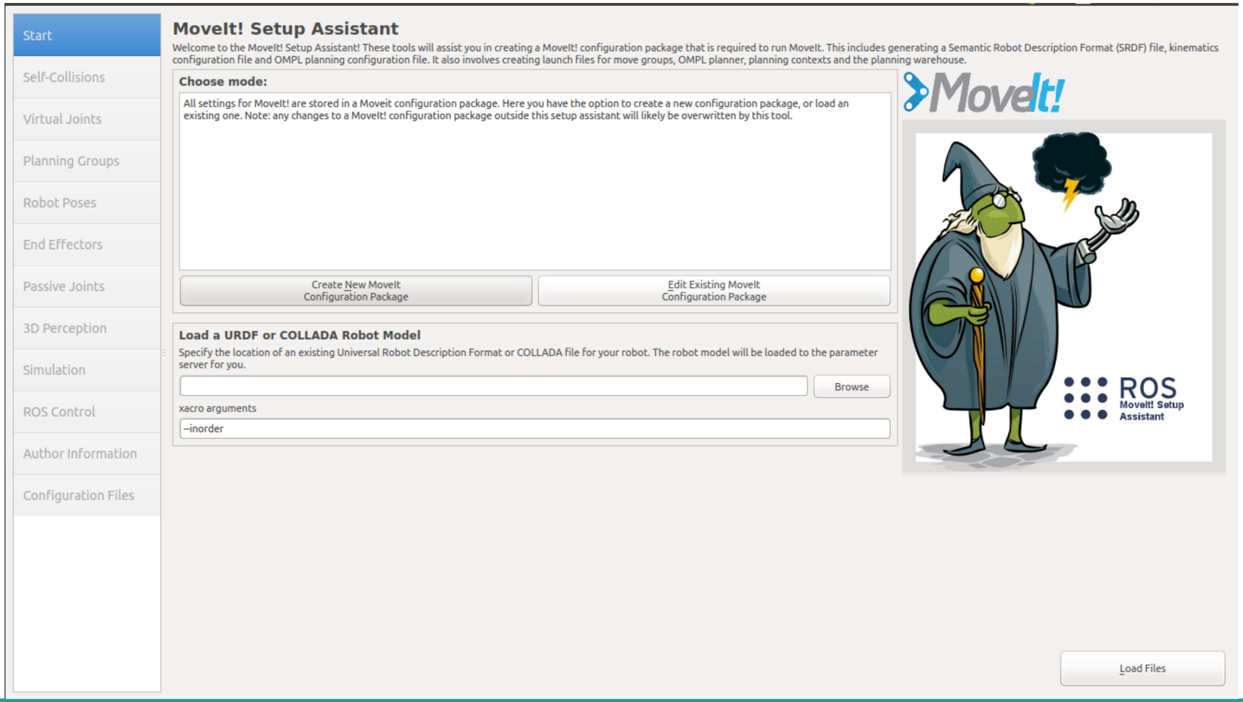
Others

- Kinamatics:
  - The default inverse kinematics plugin for MoveIt! is configured using the KDL numerical jacobian-based solver
  - Others: TRAC-IK, IKFast
- Collision Checking:
  - Collision checking in MoveIt! is mainly carried out using the FCL package

Motion Planning

MoveIt! Setup Assistant

- The MoveIt! Setup Assistant is a graphical user interface for configuring any robot for use with MoveIt!. (http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/setup_assistant/setup_assistant_tutorial.html)



AncoraSIR.com

Homework

- Modeling the Robot&Hand URDF: fill corresponding codes in [BionicDL-CobotLearning-Project1/franka_description/robots/panda_arm_hand_simulation.urdf.xacro]
- Create a MoveIt configuration package using MoveIt! Setup Assistant for franka named panda_moveit_config
- Prepare for the Project1:
  - Project1: Simulate A Picking Robot in Gazebo
  - Codes and instructions can be found at
  - https://github.com/ancorasir/BionicDL-CobotLearning-Project1
  - Brief instruction will be given at next lab

Thank you!

Prof. Song Chaoyang

- Dr. Wan Fang (sophie.fwan@hotmail.com)

AncoraSIR.com