Session 3:
Motion Control of Manipulators

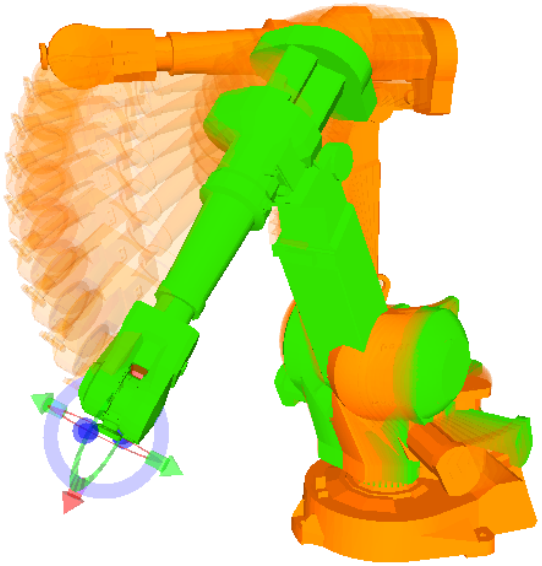
April 2015

Jeremy Zoss

Southwest Research Institute

Outline

- ROS-Industrial
  - Capabilities
  - Resources / Support
  - Architecture Overview



- Motion Planning
  - Motion Planning in ROS
  - **HowTo:** Set Up a New Robot
  - **HowTo:** Motion Planning using MoveIt!

ROS-INDUSTRIAL OVERVIEW

Industrial Robots

ABB

Motoman

Fanuc

Universal Robots

Adept

Comau*

*Coming soon

Dual Arm !

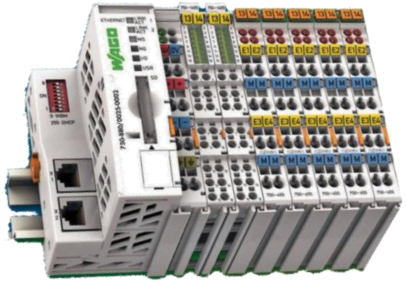
Sensor Support

Vision: Strong

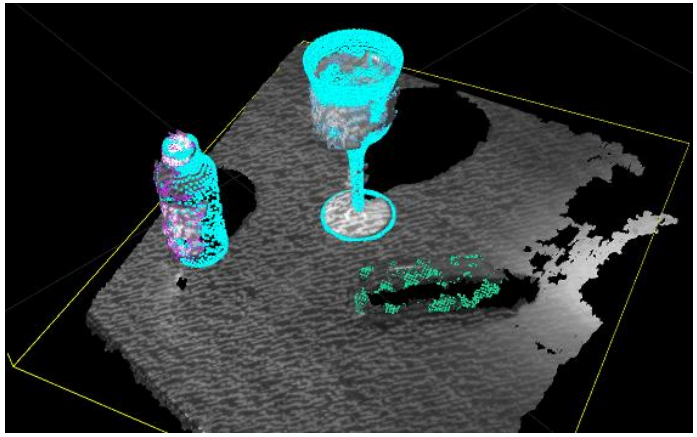
- 2D Sensors:
  - Cameras
  - Laser Scanners
- 3D Sensors
  - Stereo Vision
  - Kinect

I/O: Weak

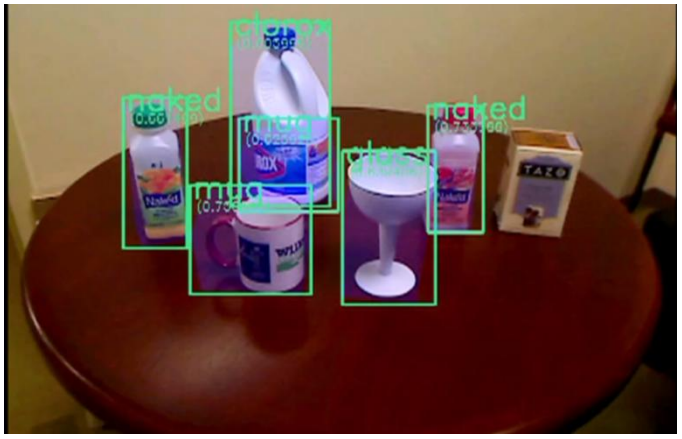
- PLC, HMI
- Network I/O
  - DeviceNet, Profibus,
  - Ethernet/IP, OPC
- Robot-Connected I/O



Perception

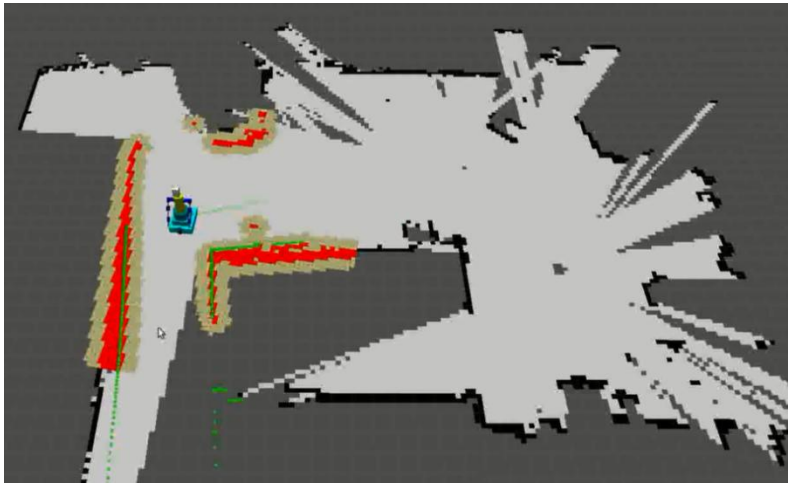


Segmentation



Identification

Mapping

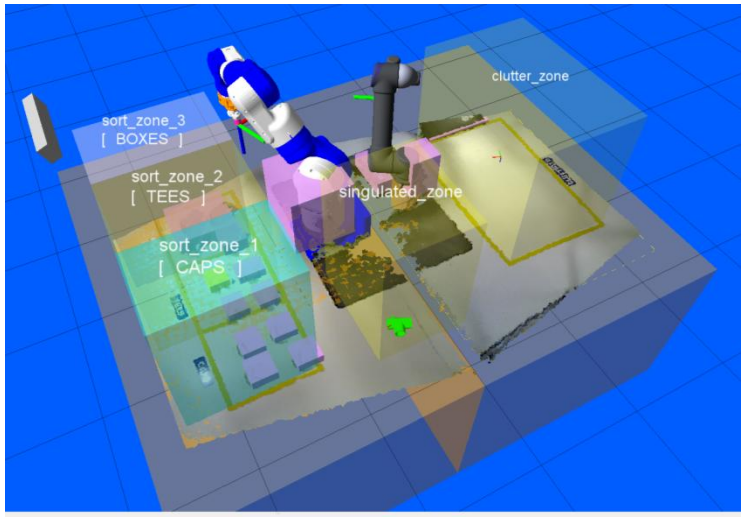


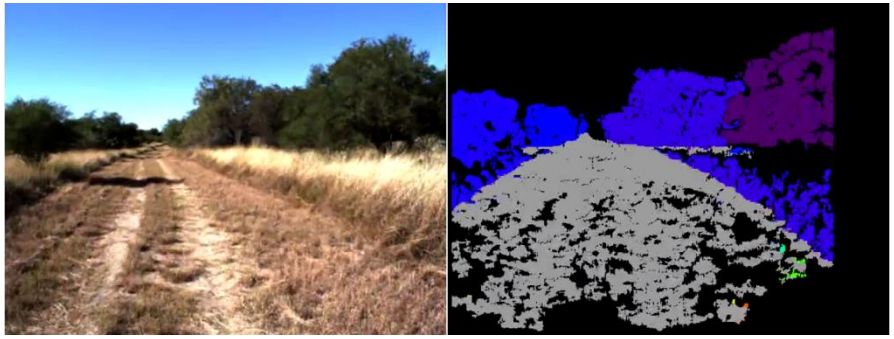
Motion Planning

- Collision-aware path planning
  - automatically finds path to target position
  - avoids obstacles, joint limits, etc.
- Can incorporate sensor data of environment

Offline Development

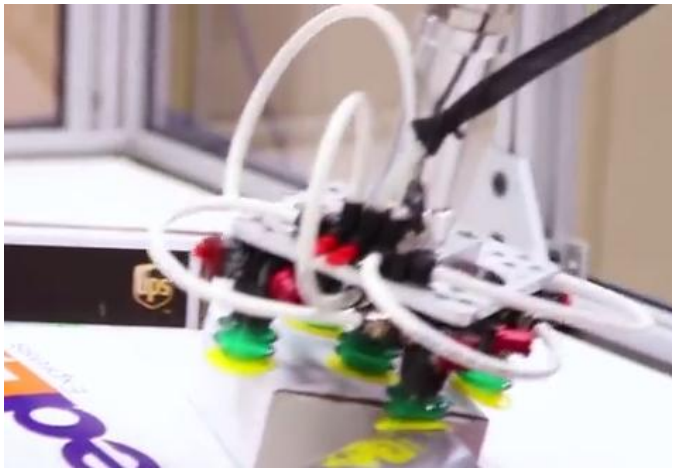
- Visualization
  - review path, debug logic
  - develop offline



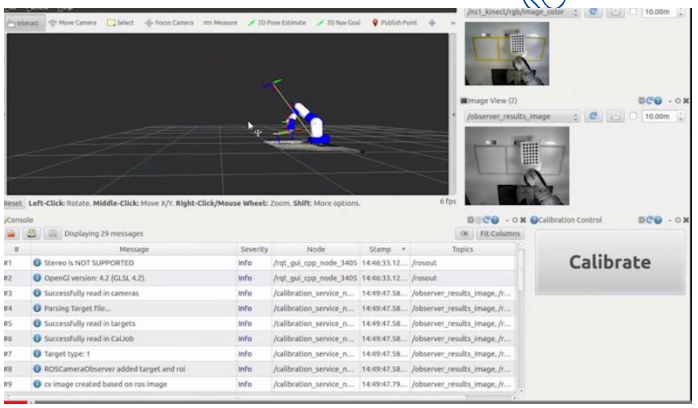


- Bag Files
  - record / playback data
  - develop offline

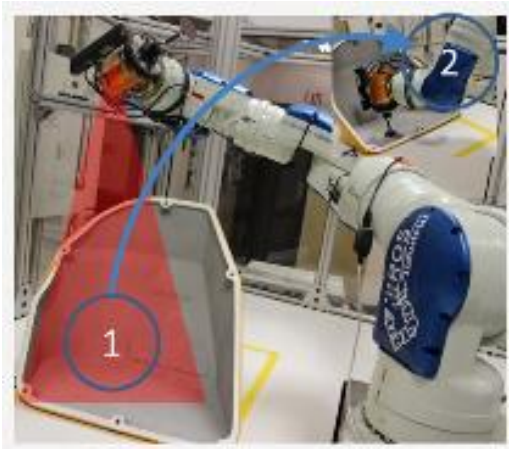
Extending ROS...



Package Singulation



Calibration



Blending (Scan/Plan)

Descartes!

Routing (Edge-Path)

ROS-I Resources

- Resources:
  - Wiki – http://wiki.ros.org/Industrial
  - Code/Bugs – http://github.com/ros-industrial
  - Email – http://groups.google.com/group/swri-ros-pkg-dev
  - Twitter – https://twitter.com/ROSIndustrial
  - Blog – http://rosindustrial.org/news/
  - YouTube channel

ROS-I Wiki

- Organized by **packages** & meta-packages
- Contains info for both **users** and **developers:**

Users

- Overview
- Tutorials

Developers

- Architecture/Design
- Code API
- Dependencies
- Bug Tracker

Getting Help

- Developers List: http://groups.google.com/group/swri-ros-pkg-dev
  - Join here

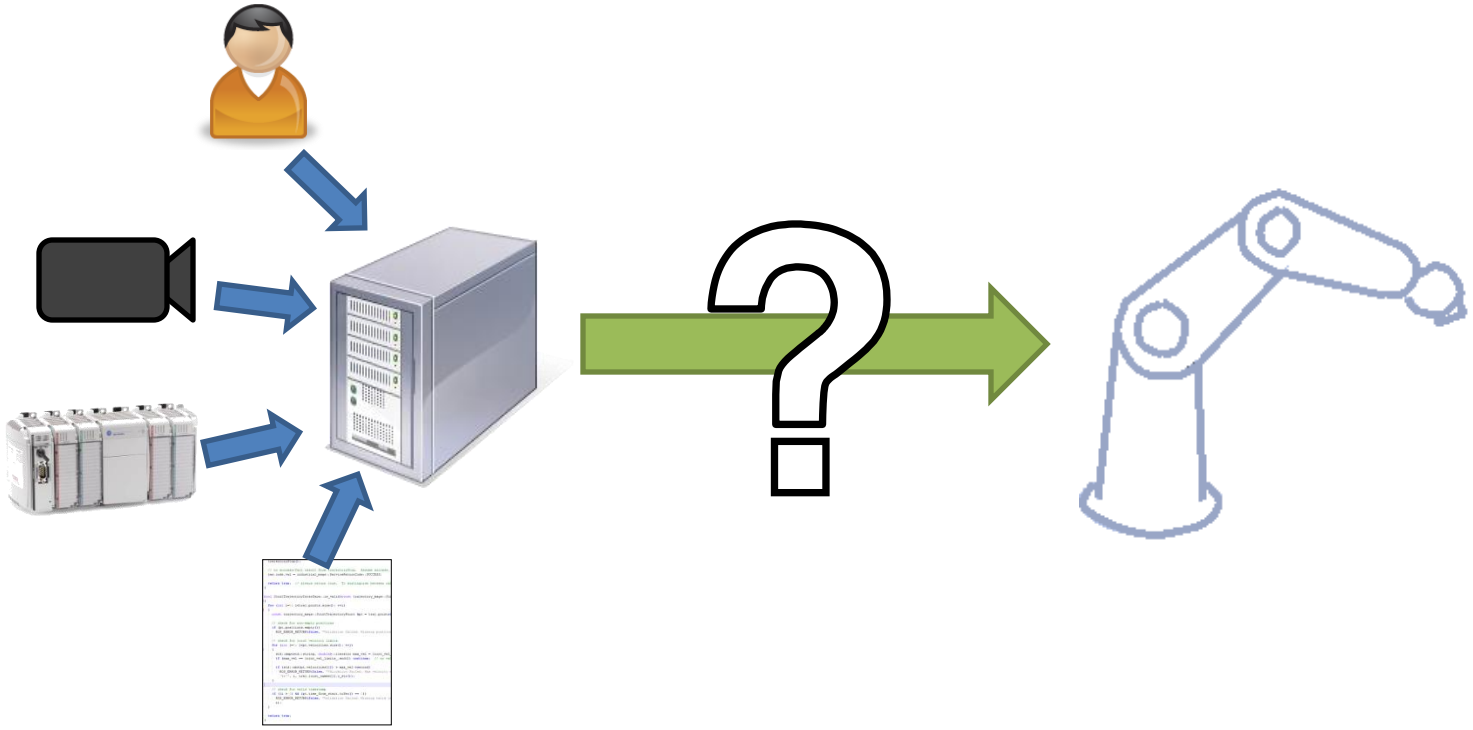
- Submitting Bugs
  - Github Code: https://github.com/ros-industrial
  - Good bug submission:
    - Setup, Steps to reproduce, Expected Results
  - Best bug submission ever:
    - Good submission + Test Code

ROS-I is a Community

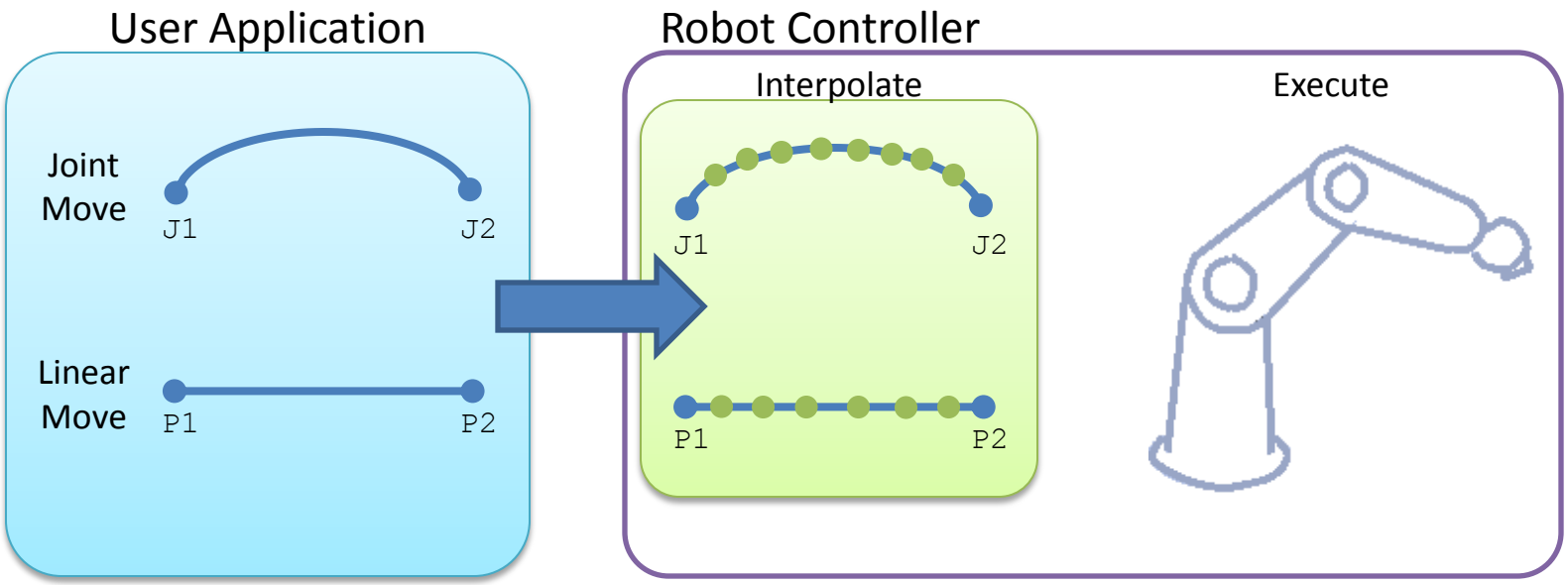
- Community is managed by SwRI & RIC
  - Monitored: Wiki, Group emails, Forums
  - Access Controlled: Repos, Twitter, Blogs, YouTube
- Consortium Members
- Public Members
  - users, developers, researchers, skeptics

Motion Planning in ROS

Motion Planning in ROS

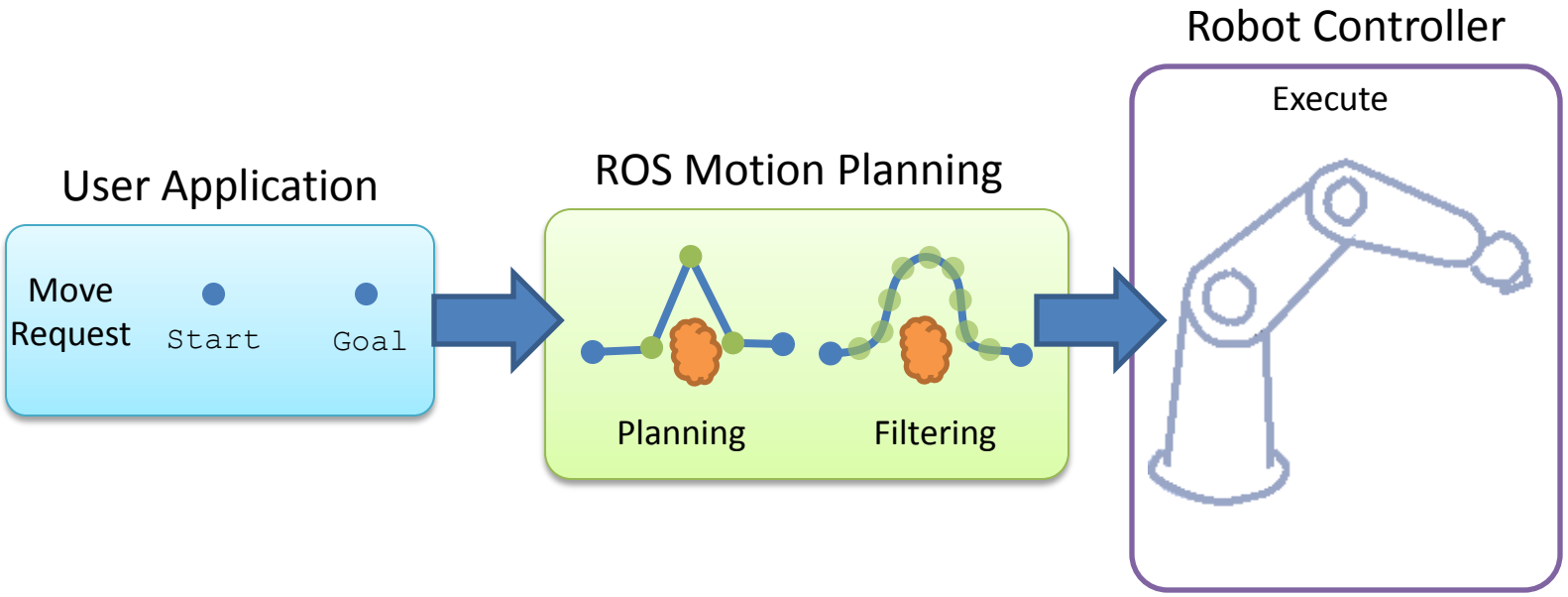


Traditional Robot Programming



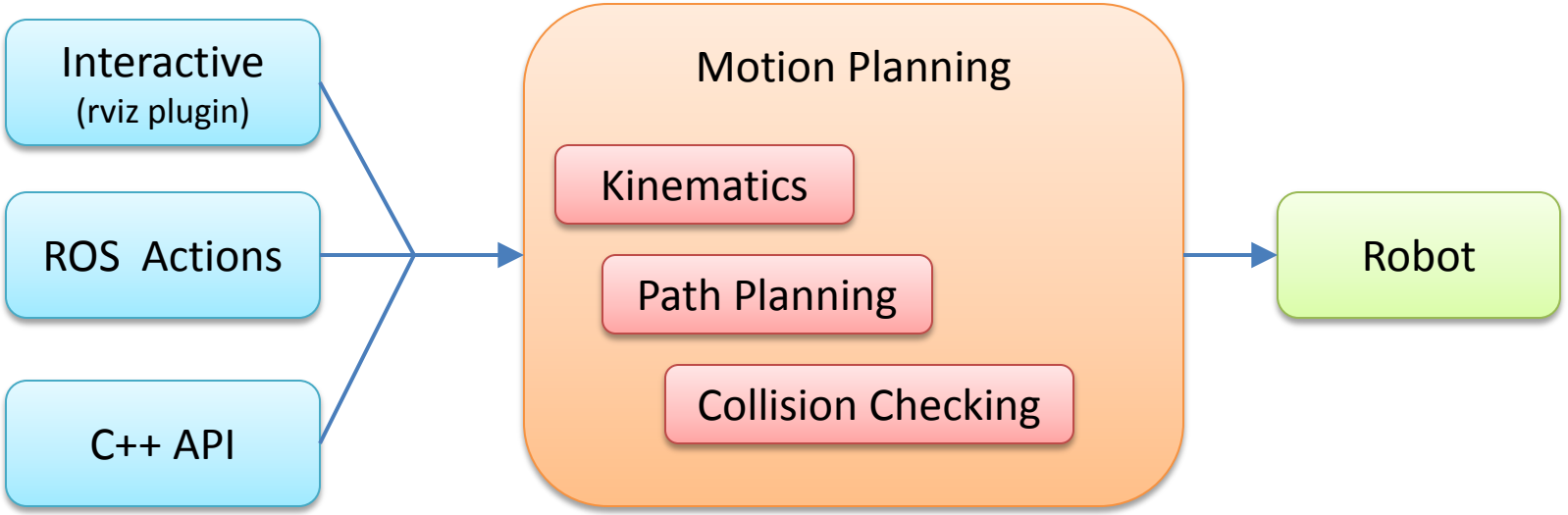
- Motion Types: *limited, but well-defined. One motion task.*
- Environment Model: *none*
- Execution Monitor: *application-specific*

ROS Motion Planning

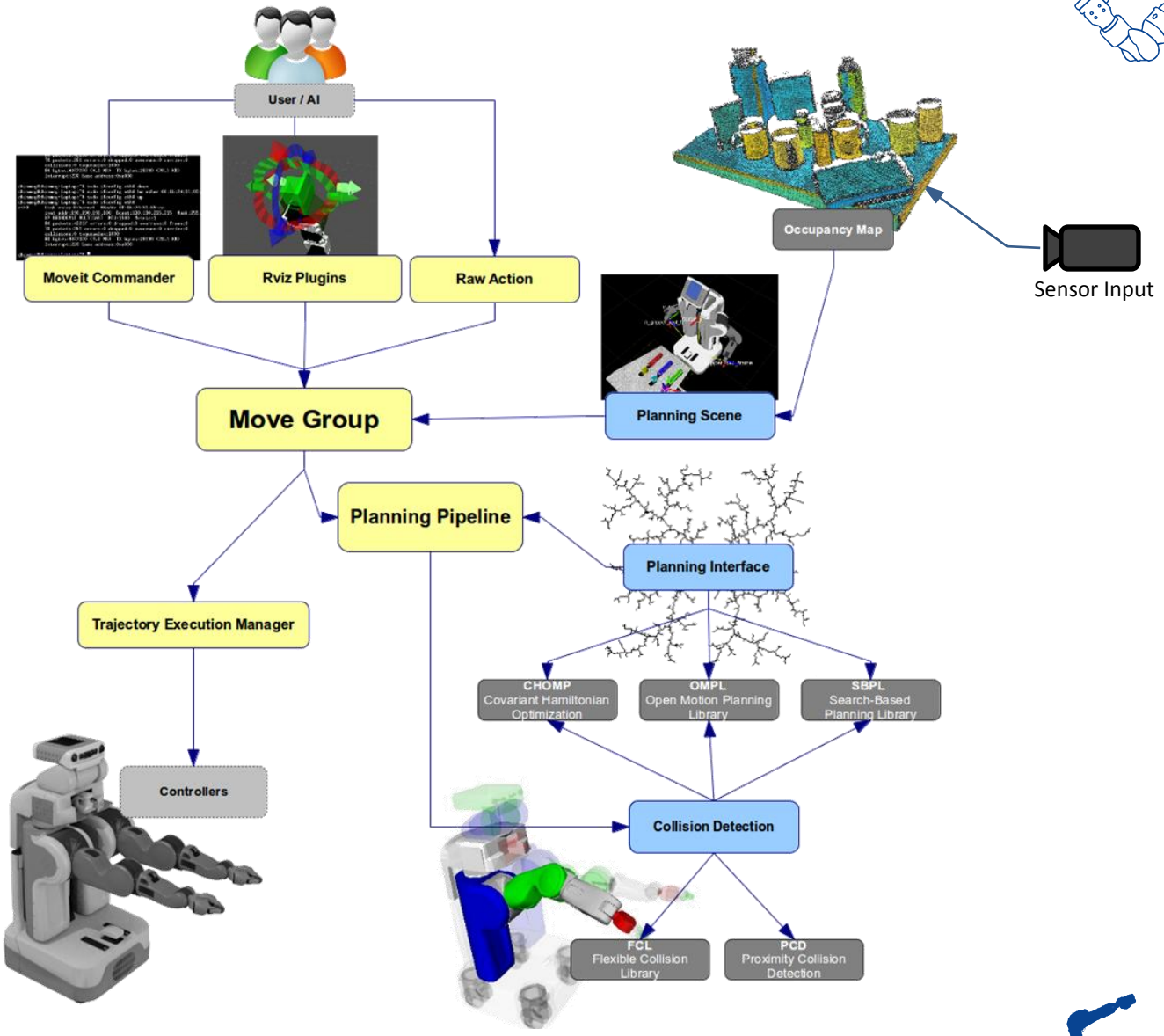


- Motion Types: *flexible, goal-driven, with constraints*
  but minimal control over actual path
- Environment Model: *automatic, based on live sensor feedback*
- Execution Monitor: *detects changes during motion*

Motion Planning Components

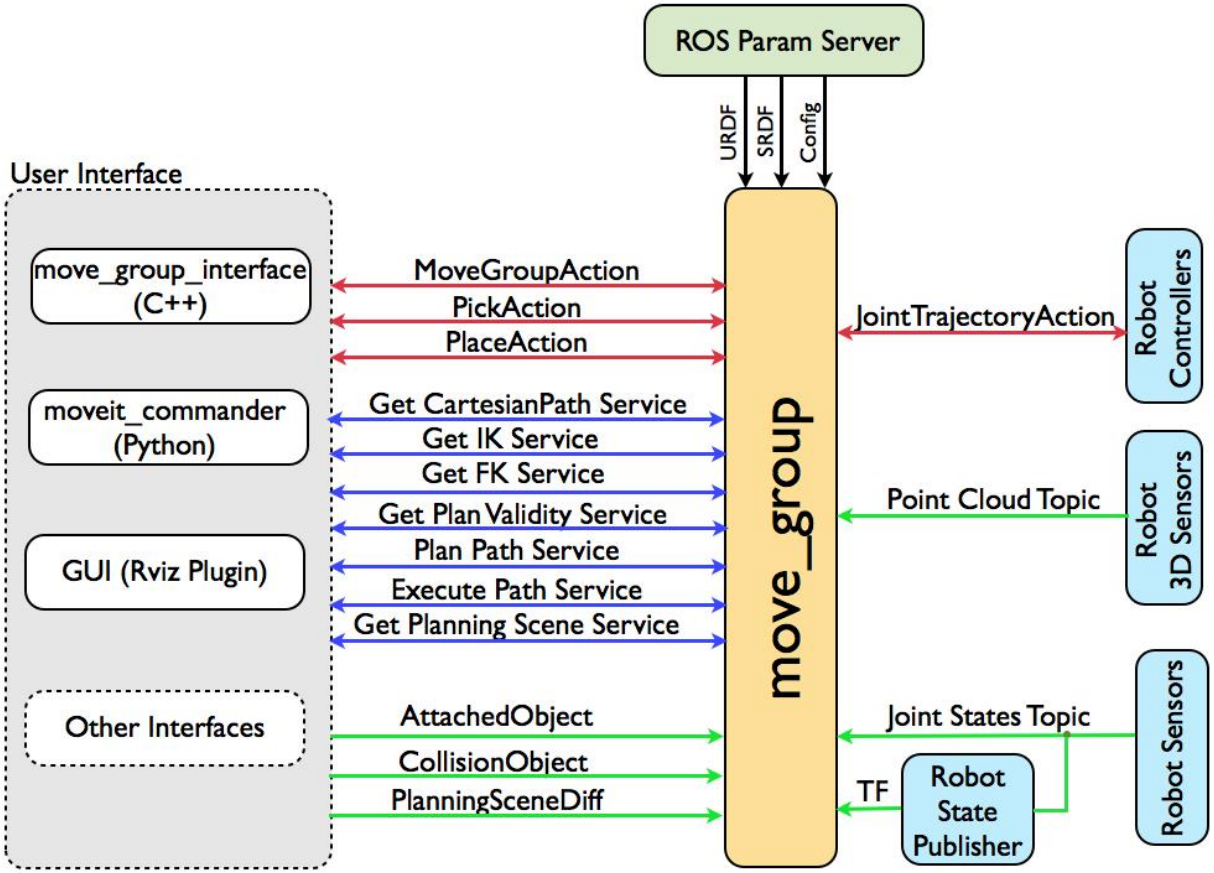


MoveIt Components



http://moveit.ros.org/wiki/High-level_Overview_Diagram
http://moveit.ros.org/wiki/Pipeline_Overview_Diagram

MoveIt Nodes



http://moveit.ros.org/documentation/concepts/

Arm Nav. vs. MoveIt!

Arm Navigation

- Released in 2011
  - deprecated in Groovy
- Multiple nodes (8)
  - *extend by replacing nodes*
- Standard ROS Interfaces
  - messages, services, actions, ...
- Somewhat slow
  - ROS comms. overhead

MoveIt!

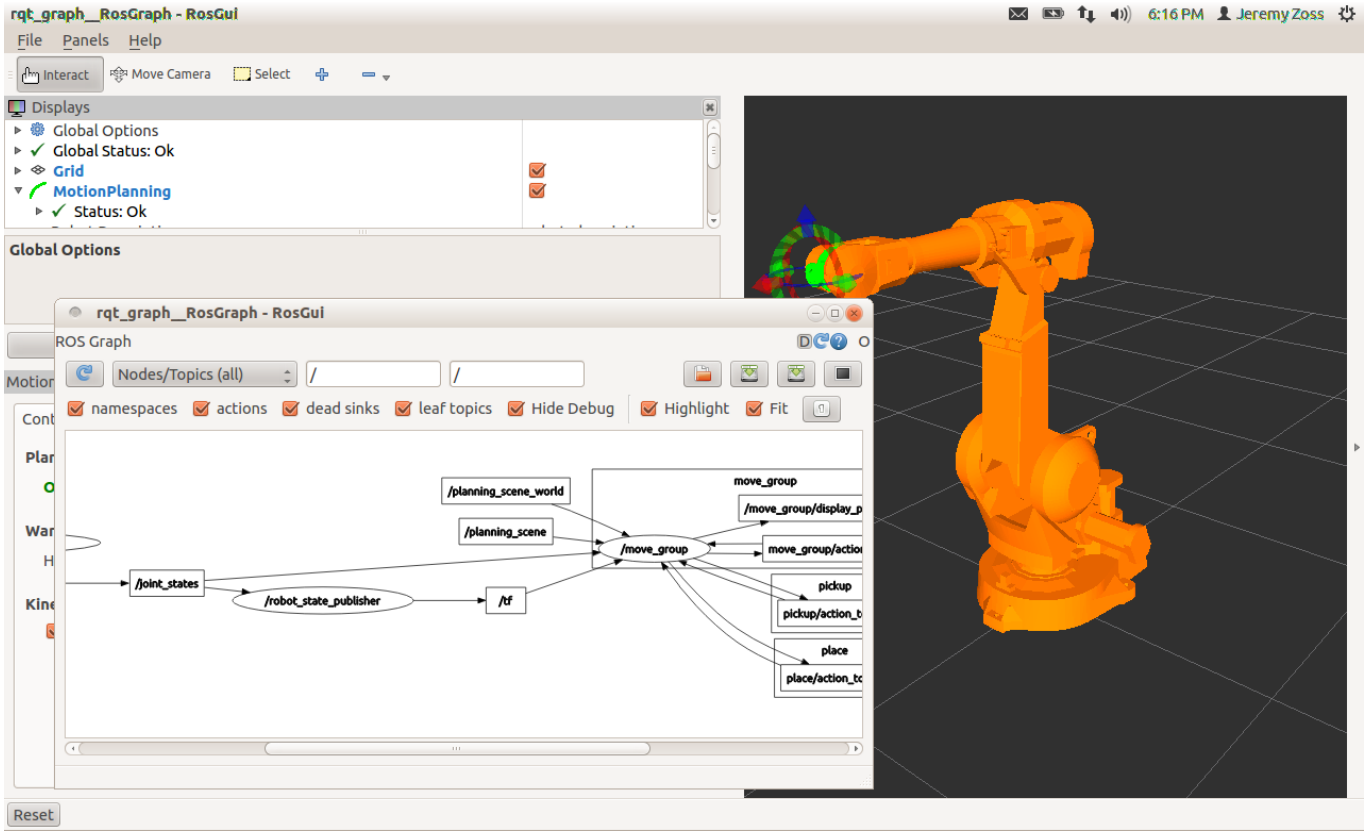
- Released in 2013
  - API still evolving
- One node
  - component plugins
  - *extend by replacing plugins*
- Multi-Interfaces
  - high-lvl API
  - C++ objects
  - messages, services, ...

MoveIt! / Robot Integration

- A MoveIt! Package...
  - includes all required nodes, config, launch files
    - motion planning, filtering, collision detection, etc.
  - is unique to each individual robot model
    - includes references to URDF robot data
  - uses a standard interface to robots
    - publish trajectory, listen to joint angles
  - can (optionally) include workcell geometry
    - e.g. for collision checking

Exercise 3.1:

Examining a Planning Environment



HowTo:

Set Up a New Robot

(or workcell)

Motivation

For each new robot model...

create a new MoveIt! package

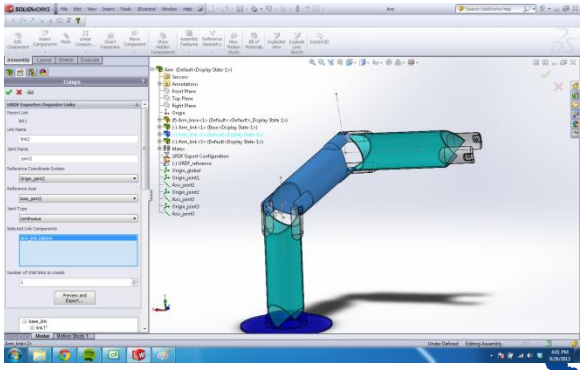
- Kinematics
  - physical configuration, lengths, etc.
- MoveIt! configuration
  - plugins, default parameter values
  - self-collision testing
  - pre-defined poses
- Robot connection
  - FollowJointTrajectory Action name

HowTo:
Set Up a New Robot

1. Create a URDF
2. Create a MoveIt! Package
3. Update MoveIt! Package for ROS-I
4. Test on ROS-I Simulator
5. Test on “Real” Robot

Create a URDF

- Previously covered URDF basics.
- Here are some tips:
  - create from datasheet or use Solidworks Add-In
  - double-check joint-offsets for accuracy
  - round near-zero offsets (if appropriate)
  - use "`base_link`" and "`tool0`"
  - use simplified collision models
    - convex-hull or primatives



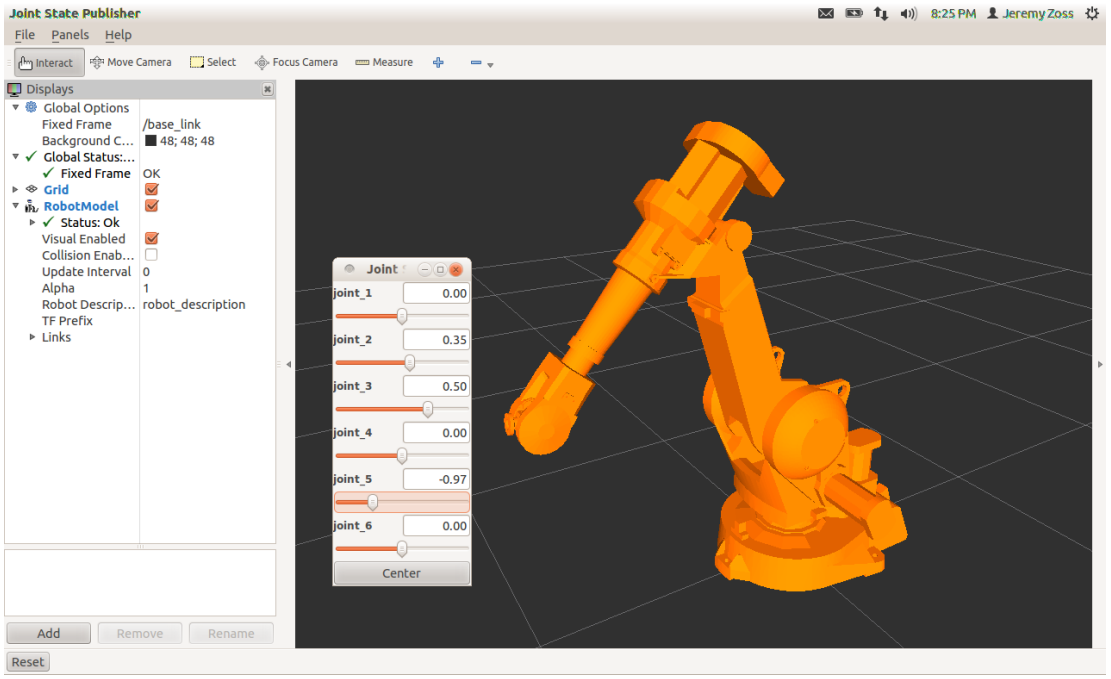
Verify the URDF

- It is **critical** to verify that your URDF matches the physical robot:
  - each joint moves as expected
  - joint-coupling issues are identified
  - min/max joint limits
  - joint directions (pos/neg)
  - correct zero-position, etc.

Exercise 3.2:

Verify the robot URDF

Watch as I demonstrate this exercise...



Create a MoveIt! Package

- Use the MoveIt! Setup Assistant
  - can create a new package or edit an existing one

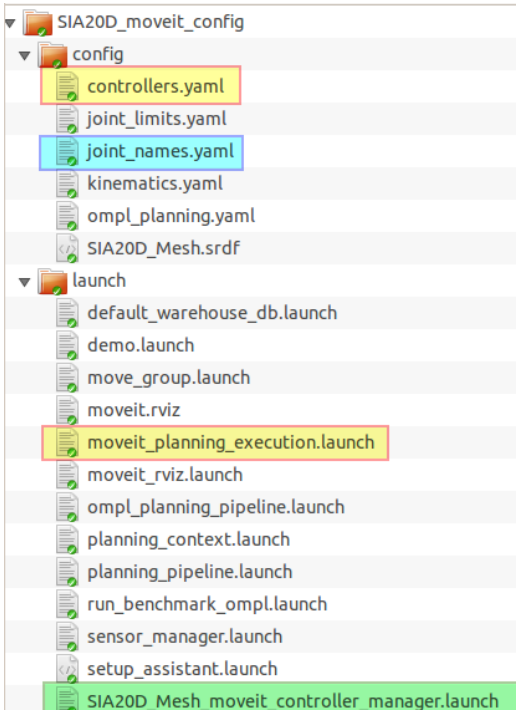
Exercise 3.3:

Create a MoveIt! Package

stop before tutorial step 2.1 (Test MoveIt Package). We'll do that later.

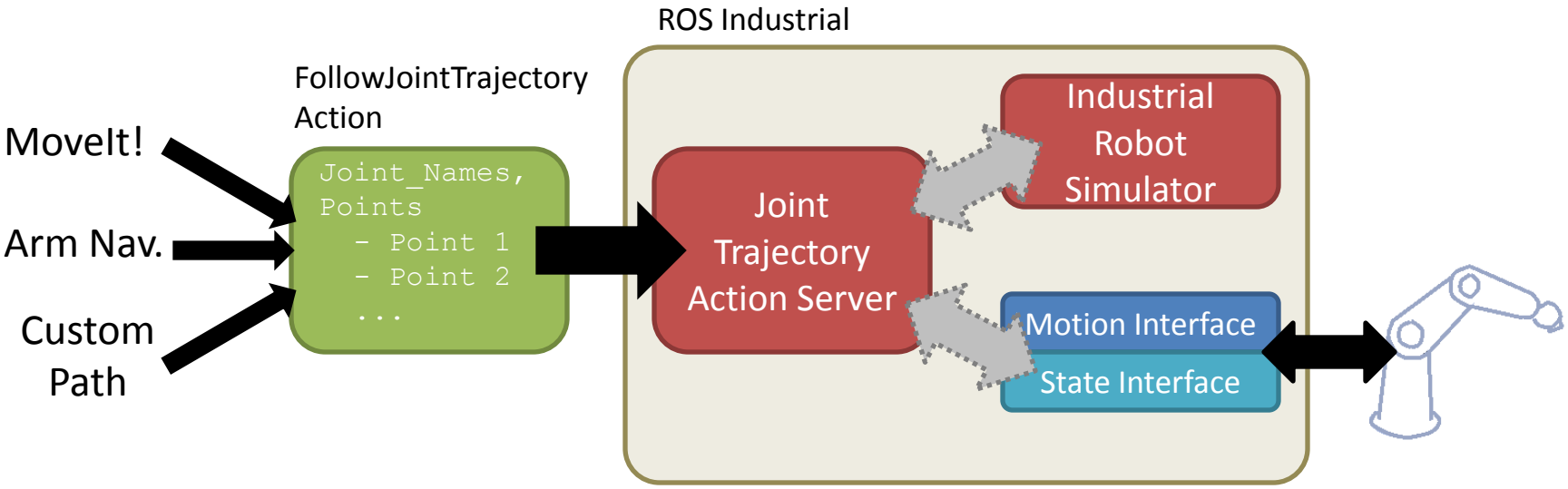
Update MoveIt! Package

- Setup Assistant generates a *generic* package
  - missing config. data to connect to a specific robot
  - ROS-I robots use a *standard* interface



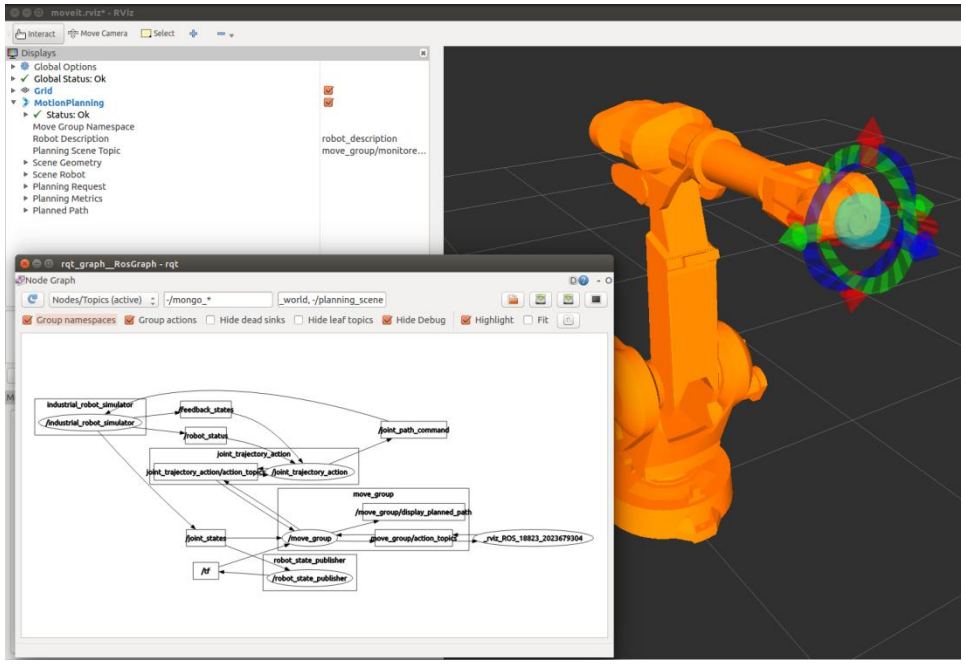
Update MoveIt! Package

- We'll generate launch files to run both:
  - **simulated** ROS-I robot
  - **real** robot-controller interface



Exercise 3.4:

Update the MoveIt! Package for ROS-I



irb2400 Substitutions:

```
[robot_moveit_config]
  => irb2400_moveit_cfg
[robot_interface_pkg]
  => abb_driver
```

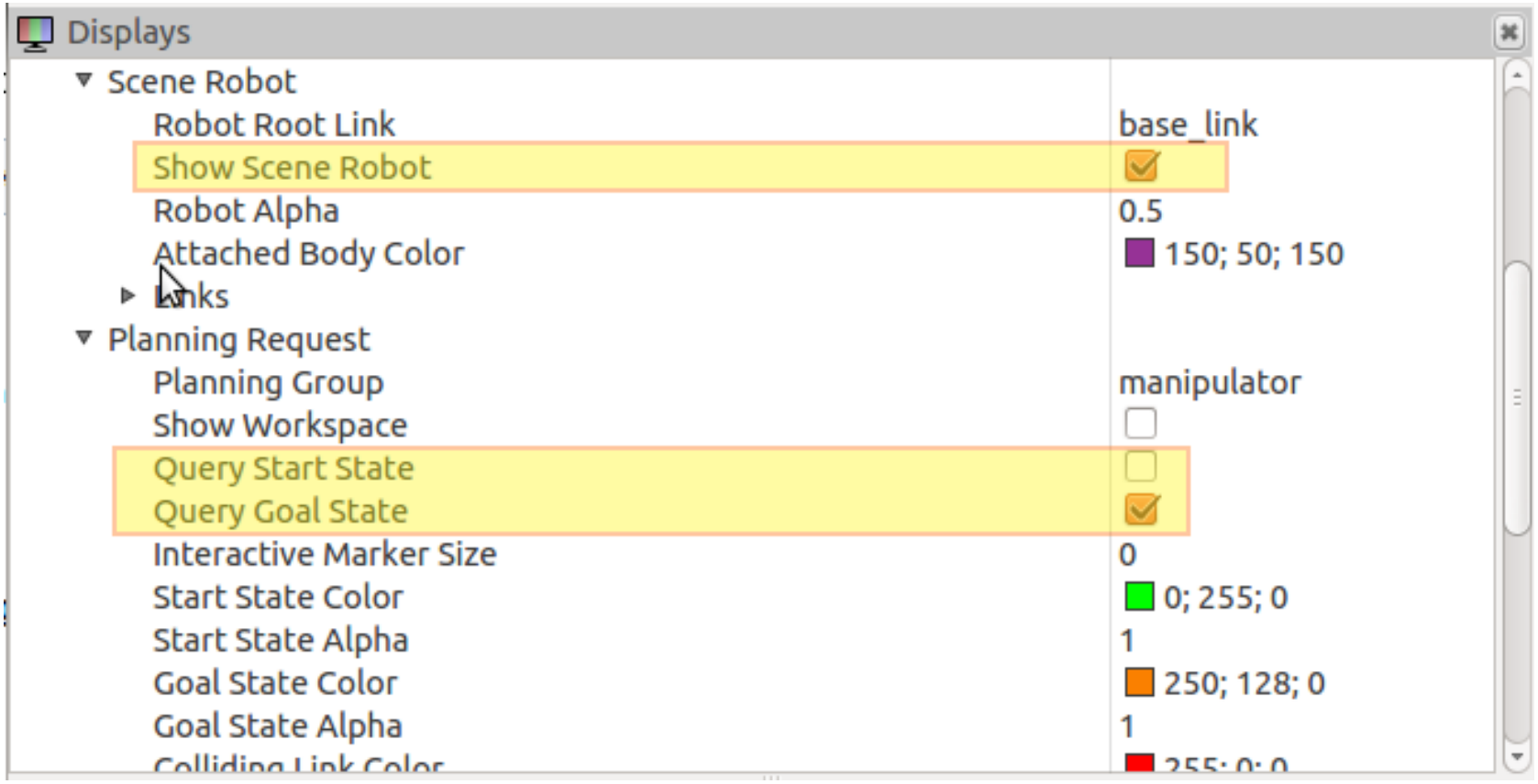
stop before tutorial step 3.1 (Test MoveIt Package). We'll do that later.

HowTo:
Motion Planning using MoveIt!

1. Motion Planning using RViz
2. Motion Planning using C++

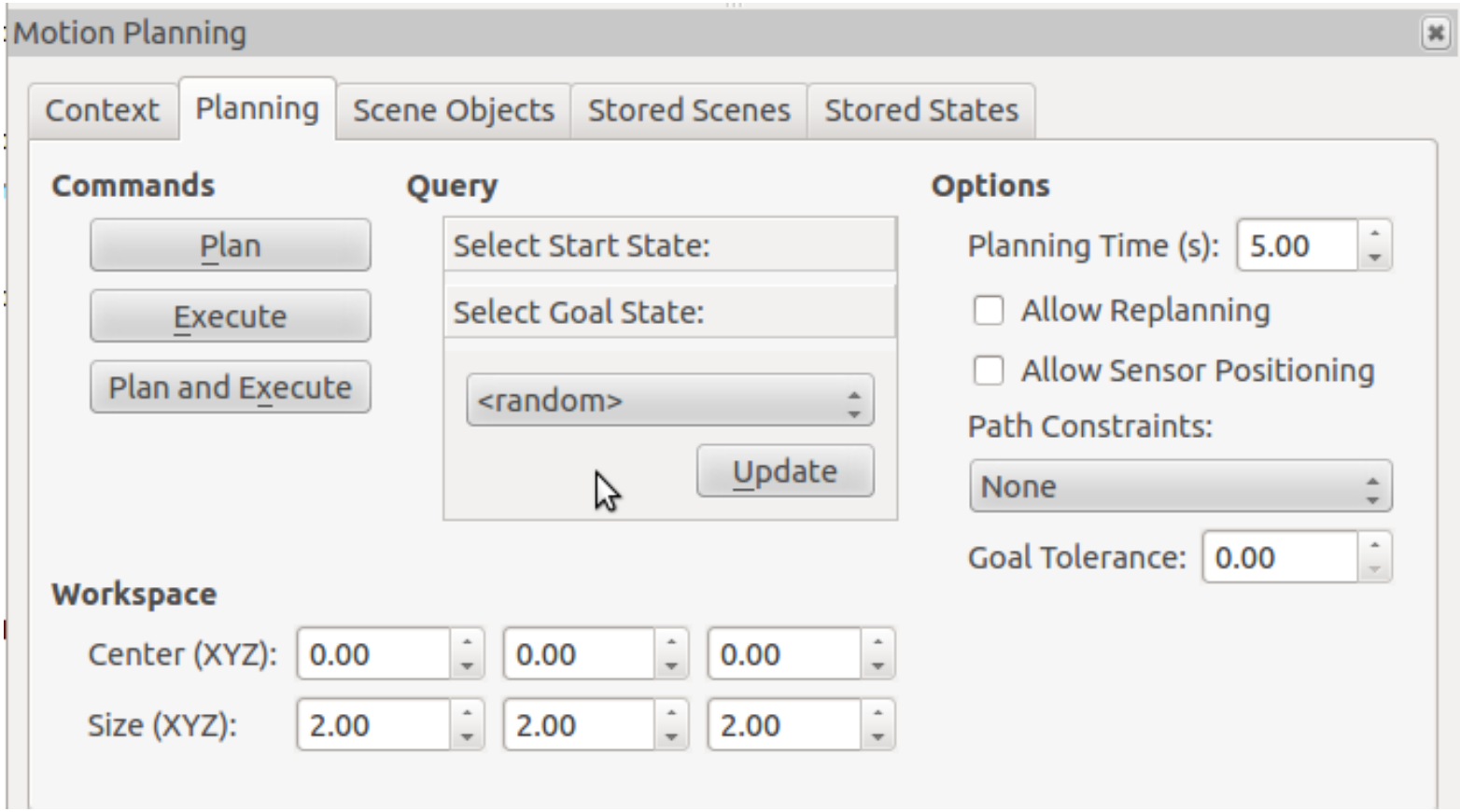
Motion Planning in RViz

Display Options



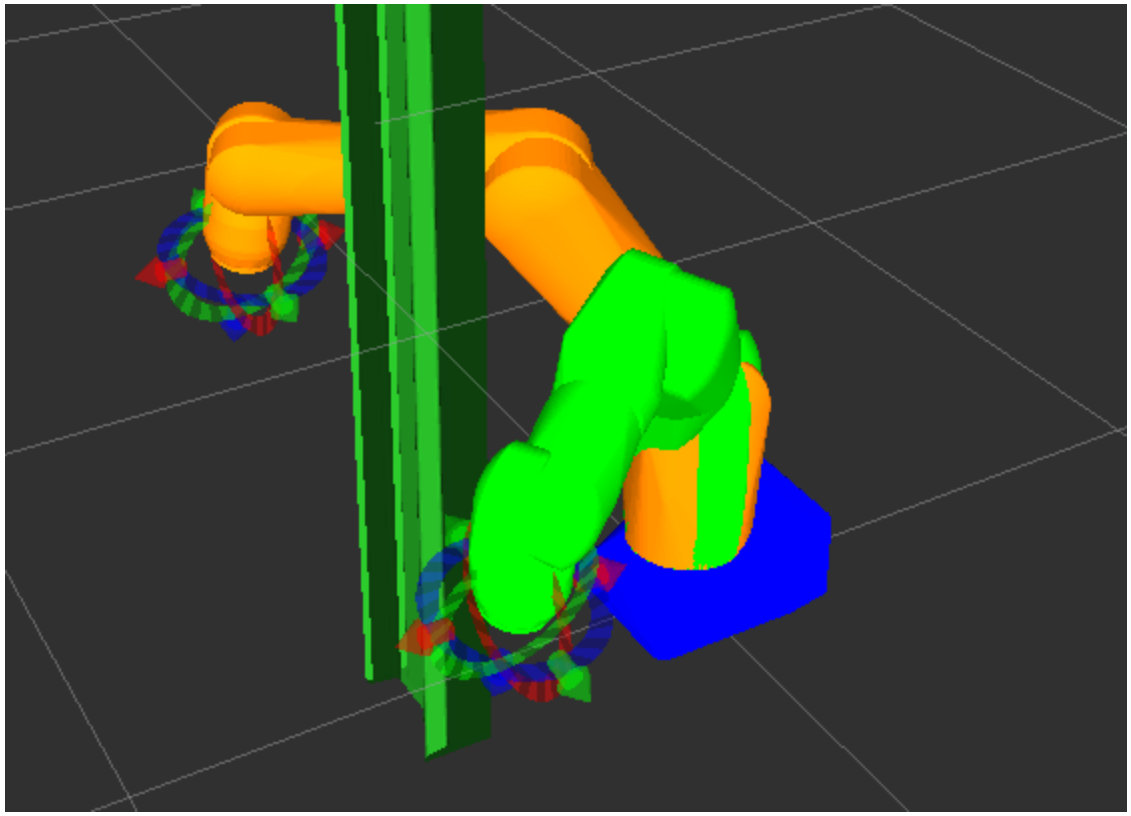
Motion Planning in RViz

Planning Options



Exercise 3.5:

Motion Planning using RViz



Motion Planning in C++

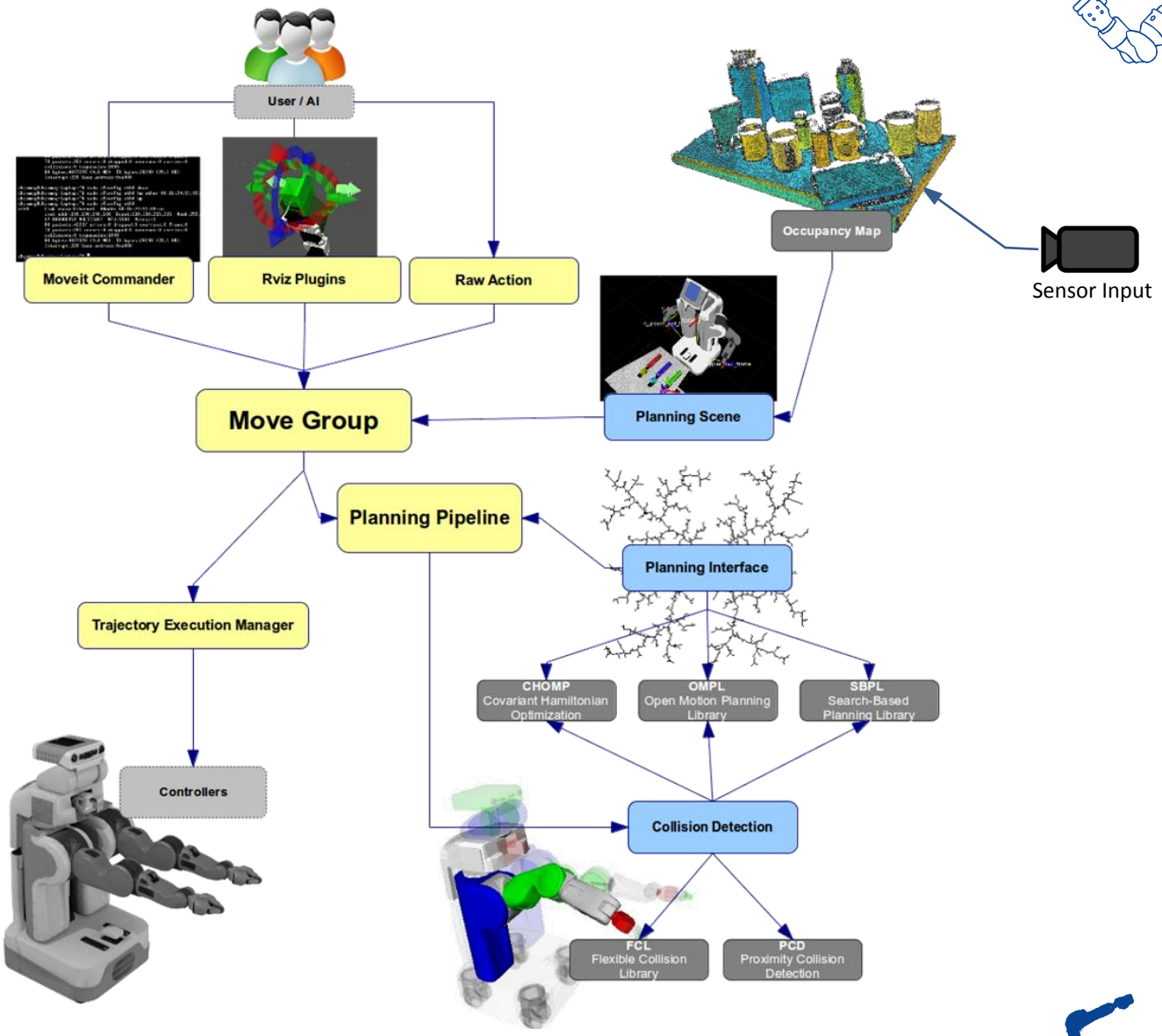
MoveIt! provides a high-level C++ API:

move_group_interface

```
#include <moveit/move_group_interface/move_group.h>
...
move_group_interface::MoveGroup group("manipulator");
group.setRandomTarget();
group.move();
```

3 lines = collision-aware path planning & execution

Reminder: MoveIt! Complexity



http://moveit.ros.org/wiki/High-level_Overview_Diagram
http://moveit.ros.org/wiki/Pipeline_Overview_Diagram

Motion Planning in C++

Pre-defined position:

```
group.setNamedTarget("home");
group.move();
```

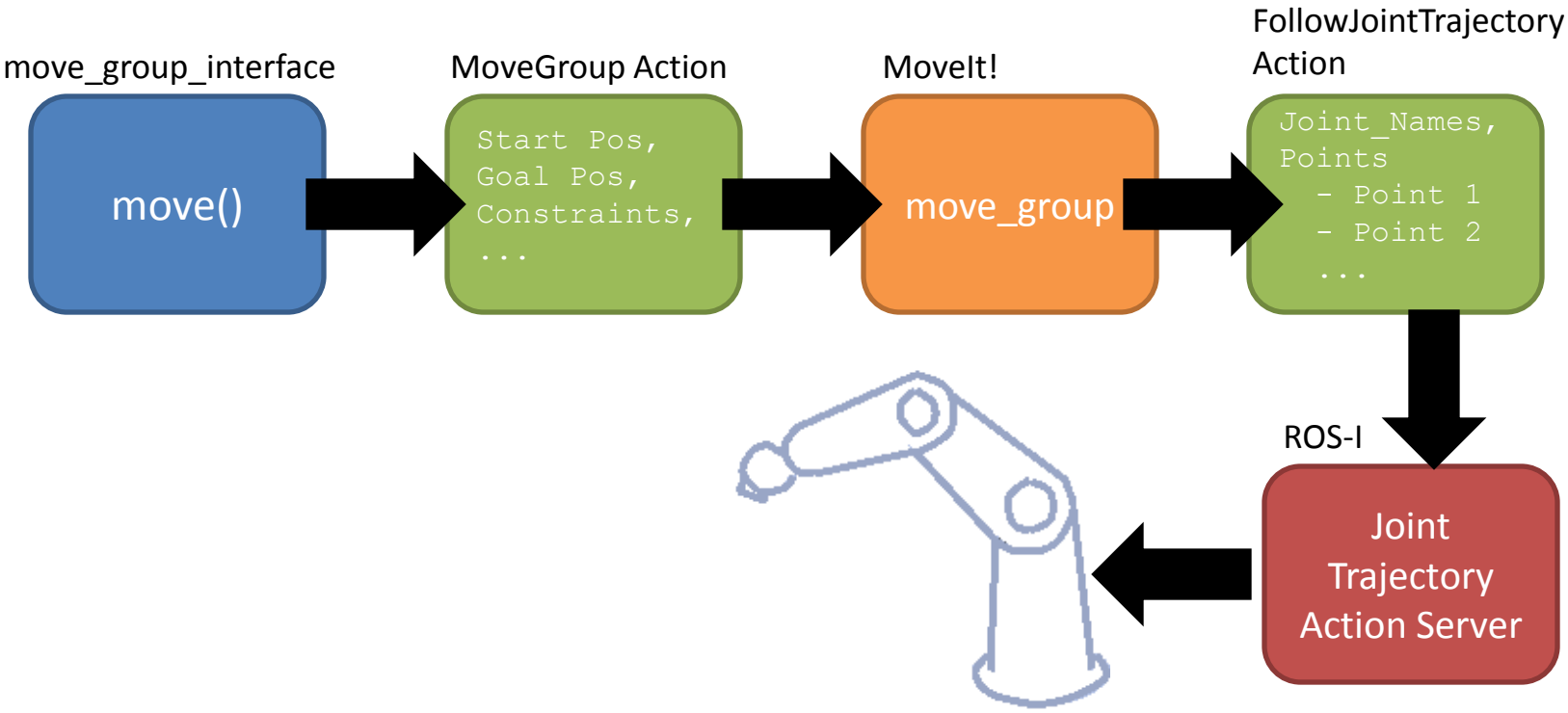
Joint position:

```
map<string, double> joints = my_function();
group.setJointValueTarget(joints);
group.move();
```

Cartesian position:

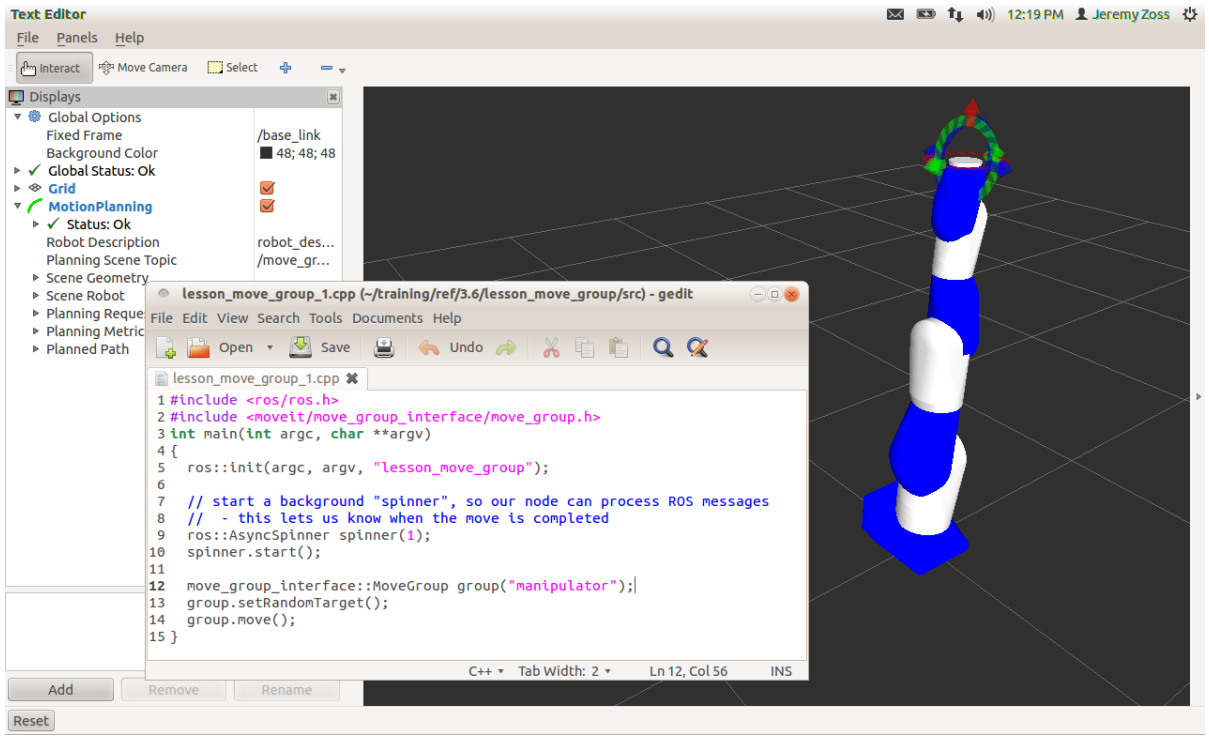
```
Affine3d pose = my_function();
group.setPoseTarget(joints);
group.move();
```

Behind the Scenes

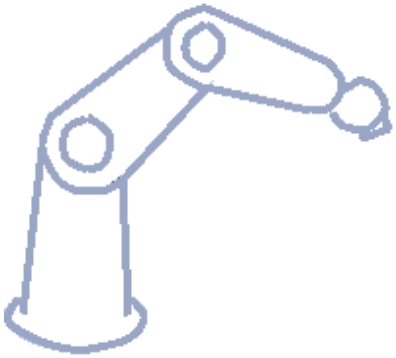


Exercise 3.6:

Motion Planning using C++



Path-to-Robot



Robot Driver / Server

- mfg-specific
- robot language
  (Karel, KRL, V+)

- feedback: pos/status
- traj pts -> motion cmd

Robot Client
ROS-I Core

- C++
- mostly standard

- robot-agnostic ROS interface

ROS Nodes

- application code
- algorithms
  user-interface

Top Level Architecture

ROS GUI
- Plugin base GUI toolkit
- Rviz, Introspection, Web-browser

ROS-I GUI (Future)
- Generic Pendant
- Standard Industrial UI

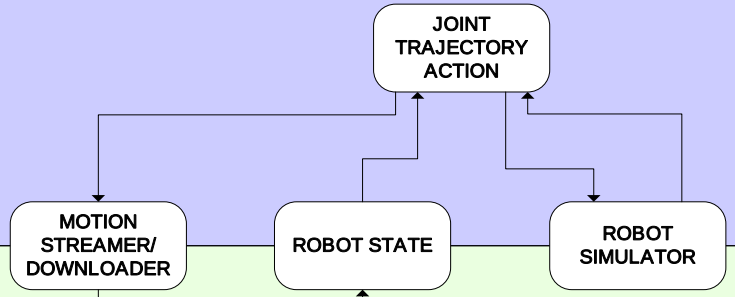
ROS Layer
- Anything in the ecosystem

MoveIt Layer
- Planning
- Kinematics
- Pick & Place
- State

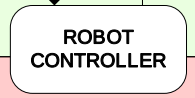
ROS-I Application Layer(Future)
- Process Planner
- State Machines

ROS-I Configuration
- urdf
- parameters
- ROS-I conventions

ROS-I Interface Layer
Package: industrial_robot_client



ROS-I Simple Message Layer
Package: simple_message

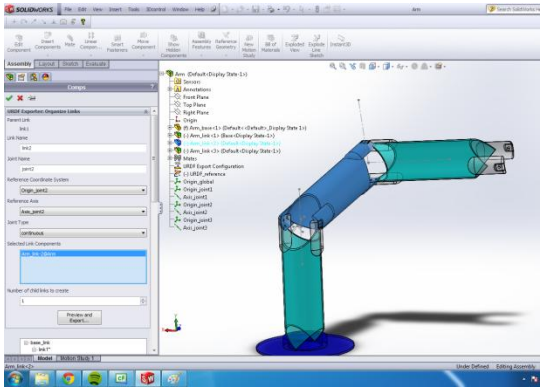


ROS-I Controller Layer
Package: vendor specific

ROS-I Vendor Configuration
- Controller options

New Robot Support

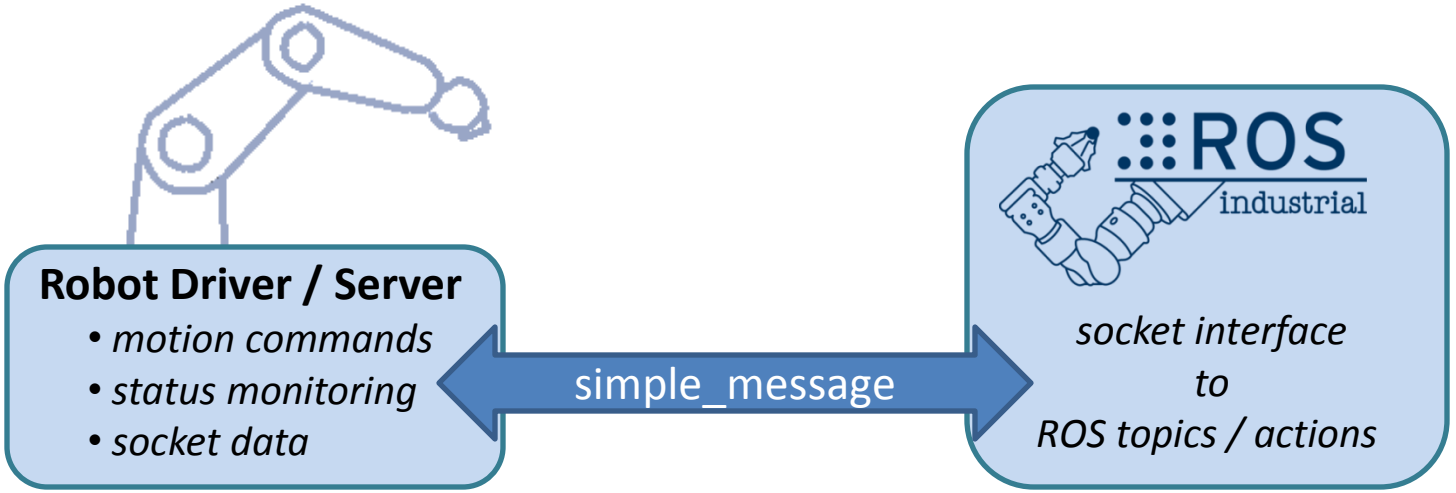
- Easily add support for new robot ***models***
  - kinematics, STL meshes
  - wizards: Solidworks, Arm Navigation
  - config only. No software changes required!



```
      <mesh filename="package://my_robot/me
    </geometry>
    <material name="abb_orange"/>
  </visual>
  <collision>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://my_robot/me
    </geometry>
    <material name="yellow"/>
  </collision>
</link>
<joint name="joint_1" type="revolute">
  <origin rpy="0 0 0" xyz="0 0 0"/>
  <parent link="base_link"/>
  <child link="link_1"/>
  <axis xyz="0 0 1"/>
```

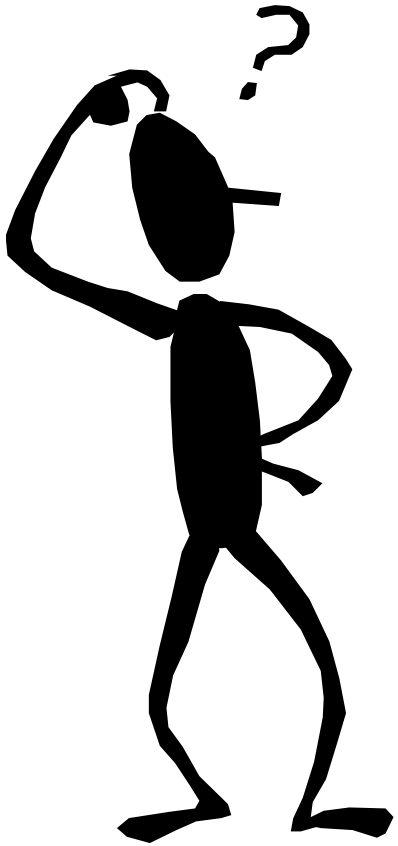
New Robot Support

- Easily add support for new robot ***models***
- Framework for new robot ***controllers***
  - robot-side driver/server
  - ROS client modifications (if needed)



Questions?

- Setup Assistant
- Robot Launch Files
- RViz Planning
- C++ Planning
- ROS-I Architecture



Contact Info.

Jeremy Zoss

Principal Engineer

SwRI
9503 W. Commerce
San Antonio, TX 78227
USA

Phone: 210-522-3089
Email: jzoss@swri.org

www.ROSindustrial.org