ROS-Industrial Basic Developer's Training Class

July 2023

Southwest Research Institute

Session 3:
Motion Control of Manipulators

Southwest Research Institute

Outline

- URDF
- TF
- Motion Planning in ROS

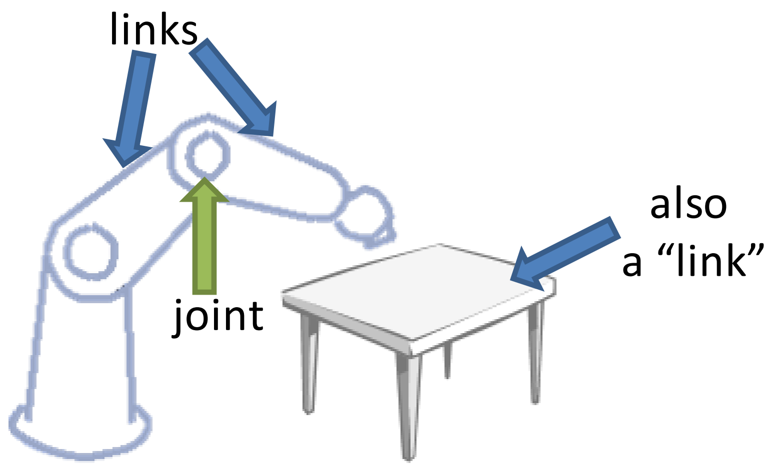
HowTo:
Set Up a New Robot

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

URDF: Unified Robot Description Format

URDF: Overview

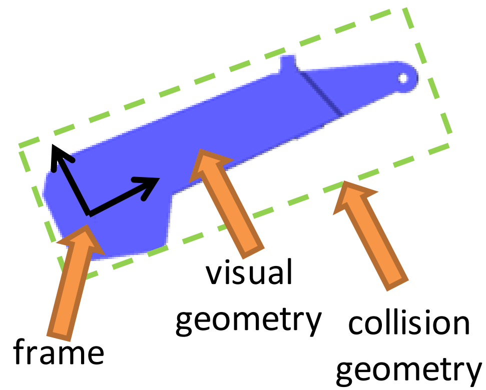
- URDF is an **XML**-formatted file containing:
  - **Links** : coordinate frames and associated geometry
  - **Joints** : connections between links
- Similar to DH-parameters (but way less painful)
- Can describe entire workspace, not just robots



URDF: Link

- A **Link** describes a physical or virtual object
  - Physical: robot link, workpiece, end-effector, ...
  - Virtual : TCP, robot base frame, ...
- Each link becomes a TF frame
- Can contain visual/collision geometry [optional]
- http://wiki.ros.org/urdf/XML/link

```
<link name="link_4">
   <visual>
        <geometry>
             <mesh filename="link_4.stl"/>
        </geometry>
        <origin xyz="0 0 0" rpy="0 0 0" />
   </visual>
   <collision>
        <geometry>
             <cylinder length="0.5" radius="0.1"/>
        </geometry>
        <origin xyz="0 0 -0.05" rpy="0 0 0" />
   </collision>
</link>
```



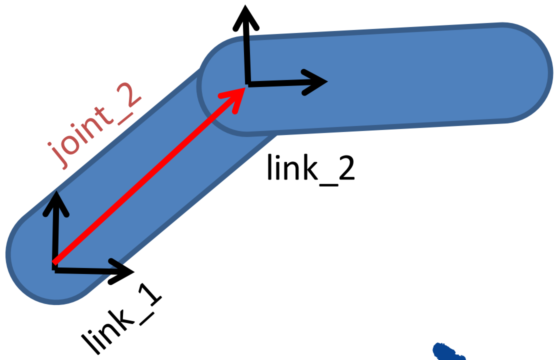
URDF Transforms

X/Y/Z	Roll/Pitch/Yaw
Meters	Radians

URDF: Joint

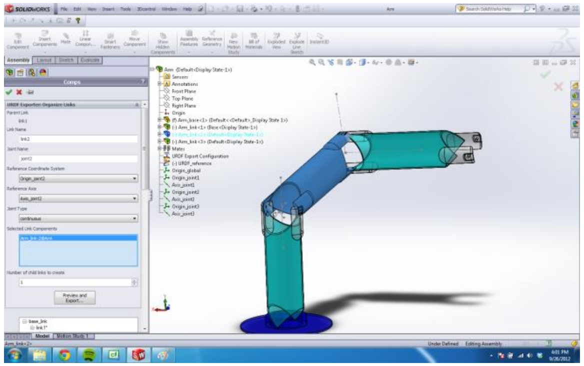
- A **Joint** connects two **Links**
  - Defines a **transform** between **parent** and **child** frames
    - Types: *fixed, free, linear, rotary*
  - Denotes axis of movement *(for linear / rotary)*
  - Contains joint limits on position and velocity
- ROS-I conventions
  - X-axis front, Z-Axis up
  - Keep all frames similarly rotated when possible
- http://wiki.ros.org/urdf/XML/joint

```
<joint name="joint_2" type="revolute">
   <parent link="link_1"/>
   <child link="link_2"/>
   <origin xyz="0.2 0.2 0" rpy="0 0 0"/>
   <axis xyz="0 0 1"/>
   <limit lower="-3.14" upper="3.14" velocity="1.0"/>
</joint>
```



URDF 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 primitives

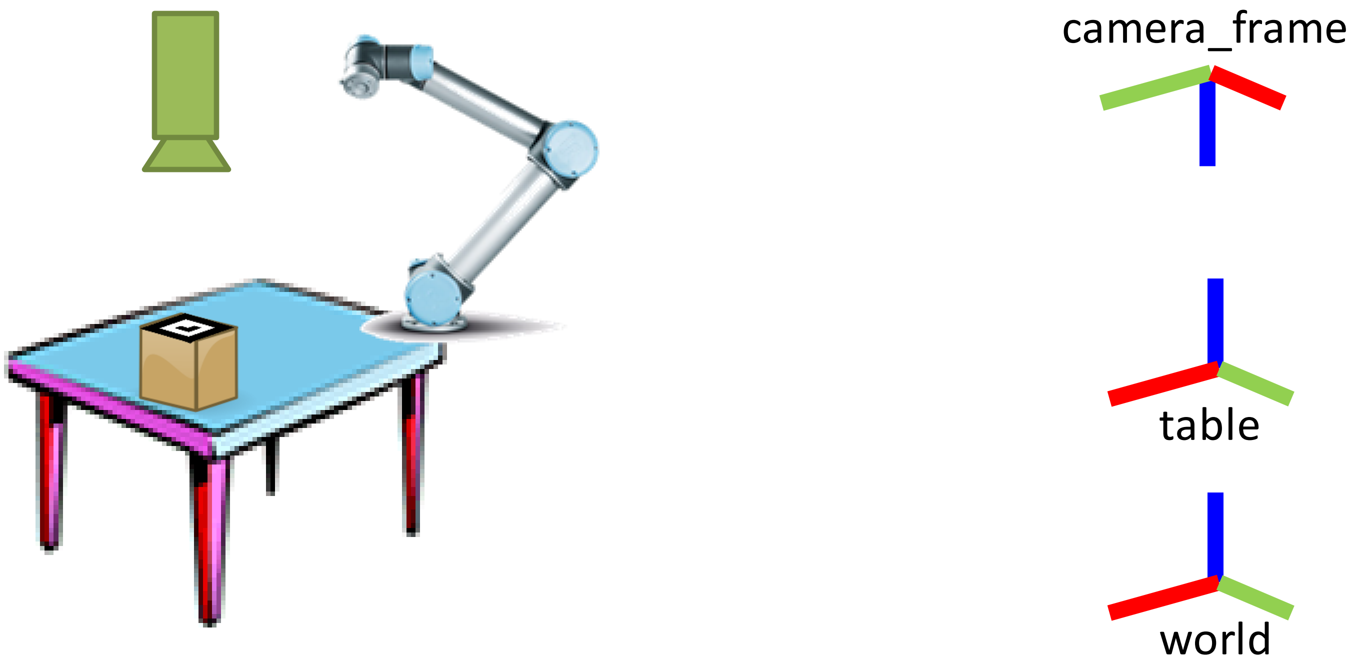


Verify the URDF

- 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.
  - check forward kinematics

Exercise 3.0

Create a simple urdf



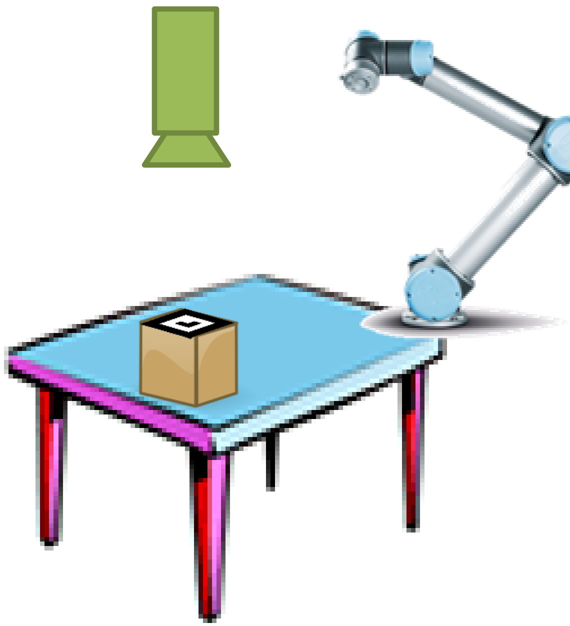
URDF: XACRO

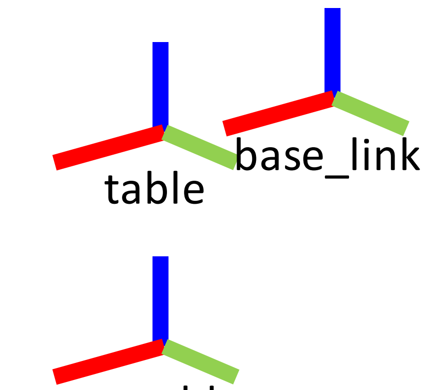
- **XACRO** is an XML-based “macro language” for building URDFs
  - <Include> other XACROs, with parameters
  - Simple expressions: math, substitution
- Used to build complex URDFs
  - multi-robot workcells
  - reuse standard URDFs (e.g. robots, tooling)

```
<xacro:include filename="myRobot.xacro"/>

<xacro:myRobot prefix="left_"/>
<xacro:myRobot prefix="right_"/>

<property name="offset" value="1.3"/>

<joint name="world_to_left" type="fixed">
    <parent link="world"/>
    <child link="left_base_link"/>
    <origin xyz="${offset/2} 0 0" rpy="0 0 0"/>
</joint>
```

Exercise 3.1

Combine simple urdf with ur5 xacro

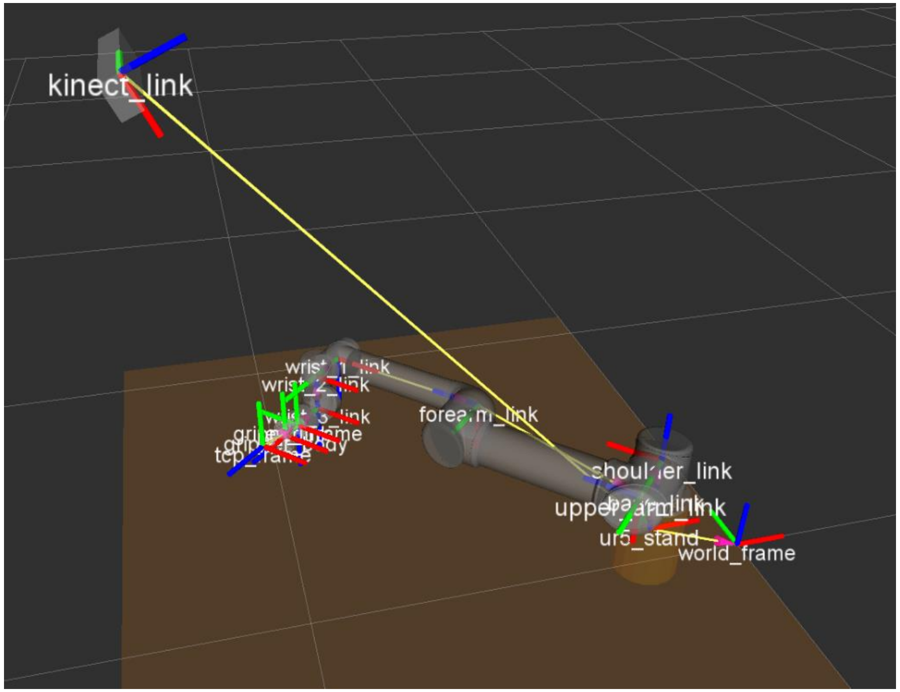




TF – Transforms in ROS

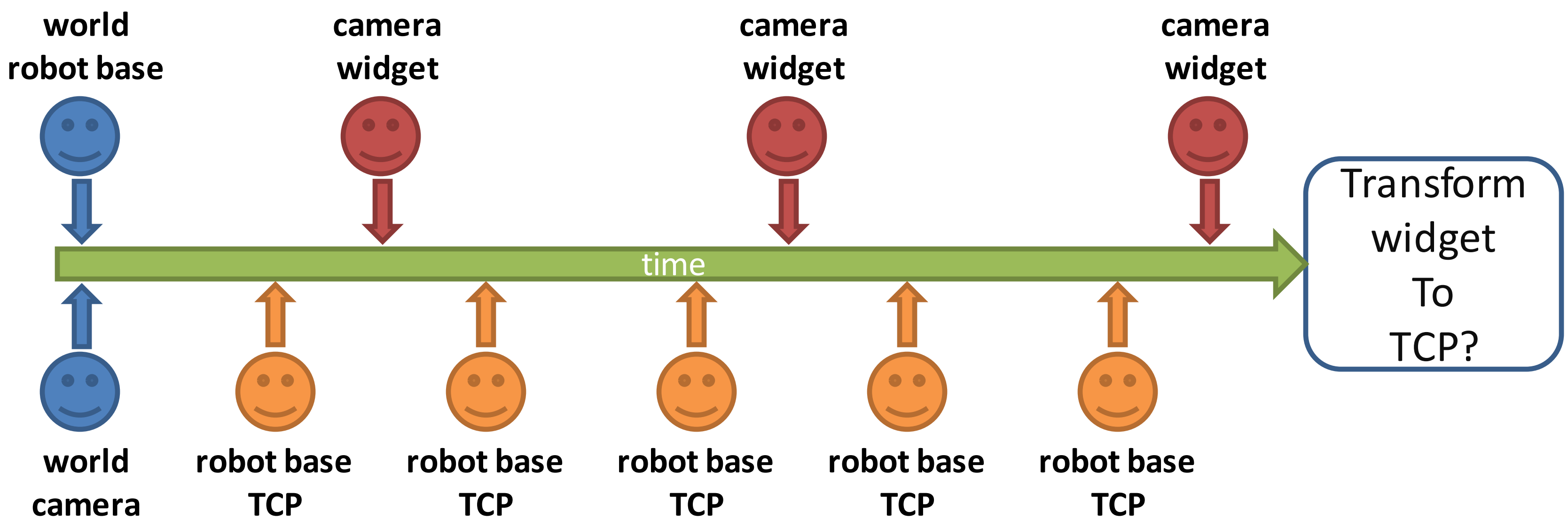
TF: Overview

- TF is a **distributed framework** to track **coordinate frames**
- Each frame is related to at least one other frame



TF: Time Sync

- TF tracks frame history
  - can be used to find transforms in the past!
  - essential for asynchronous / distributed system



TF: c++

- Each **node** has its own `TransformListener`
  - listens to <u>all</u> TF messages, calculates relative transforms
  - Can try to transform in the past
  - ➢ Can only look as far back as it has been running

```
tf2_ros::Buffer buffer(node->get_clock());
tf2_ros::TransformListener listener(buffer);

geometry_msgs::msg::TransformStamped transform;
transform = buffer.lookupTransform("target", "source", tf2::TimePointZero);
```

Result

Parent Frame ("reference")

Child Frame ("object")

Time

- Note confusing "target/source" naming convention
- Tf2::TimePointZero gives **latest** available transform

TF Timing

- When requesting a transform, you must specify a **time**:
  - Latest Received

    ```
    lookupTransform(“from”, “to”, tf2::TimePointZero)
    ```

  - Current Time (will probably fail)

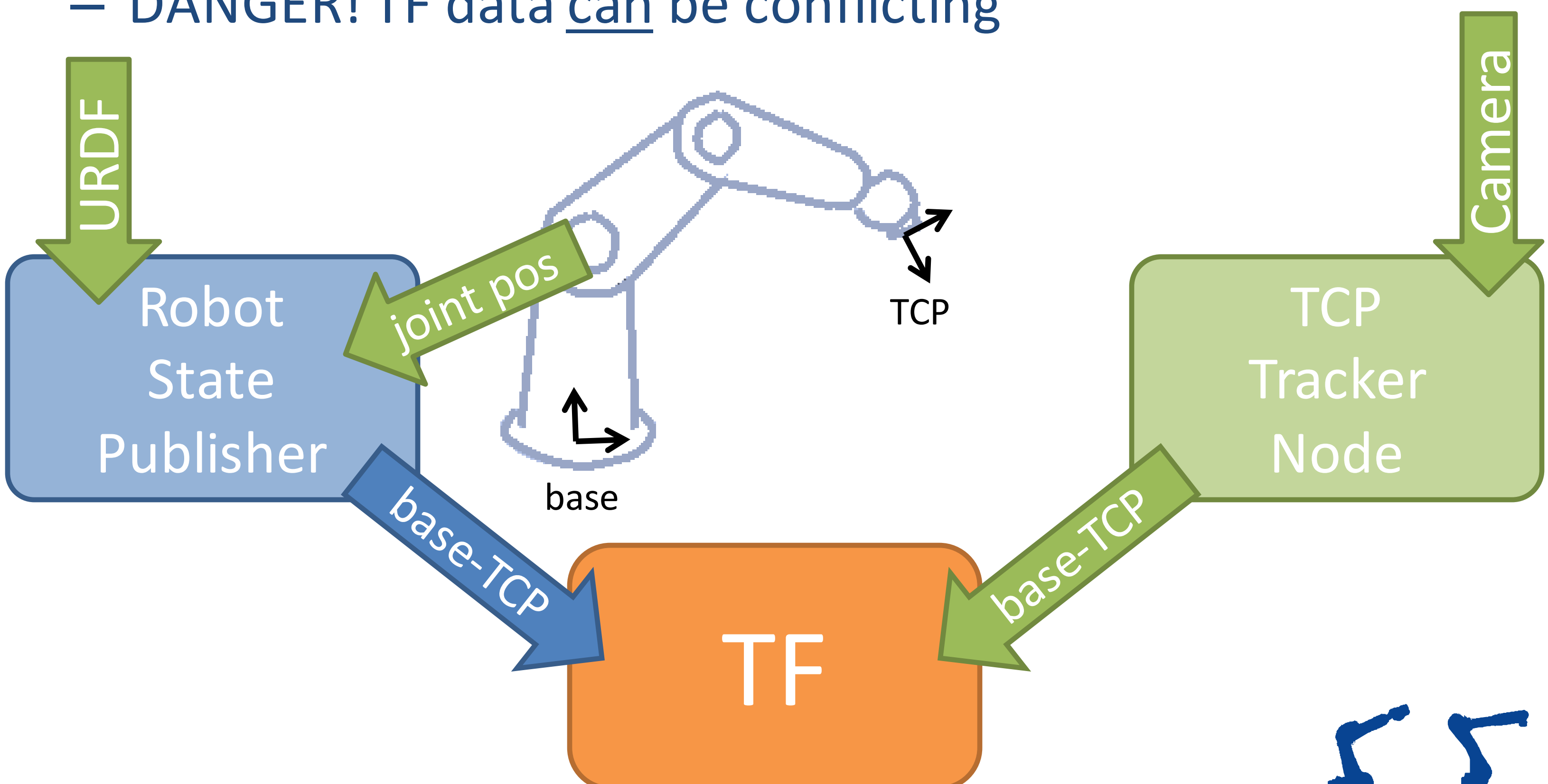
    ```
    lookupTransform(“from”, “to”, now)
    ```

  - Current Time (wait for it to be available)

    ```
    lookupTransform(“from”, “to”, now, 50ms)
    ```

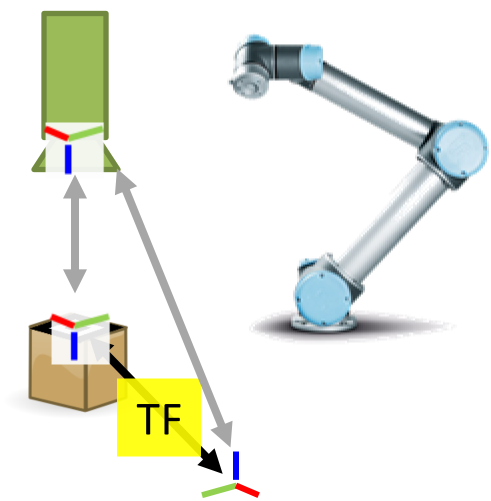
TF: Sources

- A `robot_state_publisher` provides TF data from a **URDF**
- Nodes can also publish TF data
  - DANGER! TF data can be conflicting

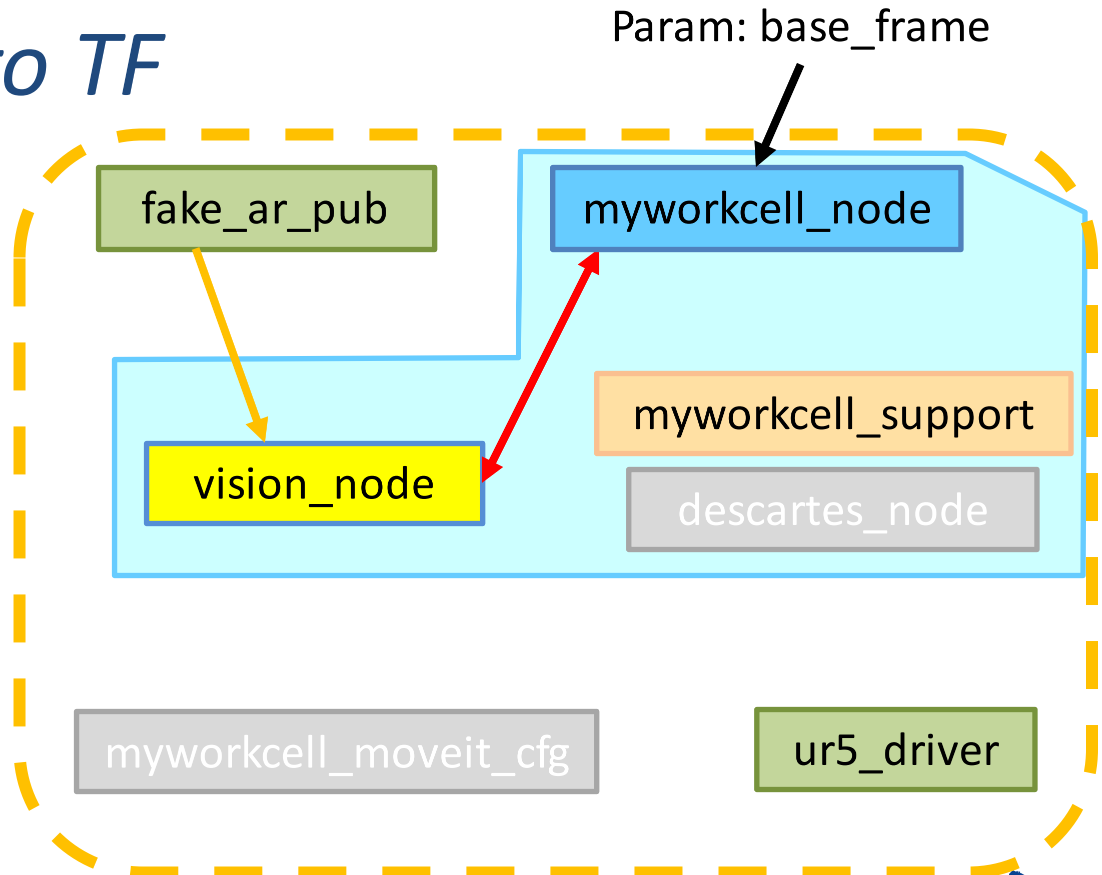


Exercise 3.2

Introduction to TF



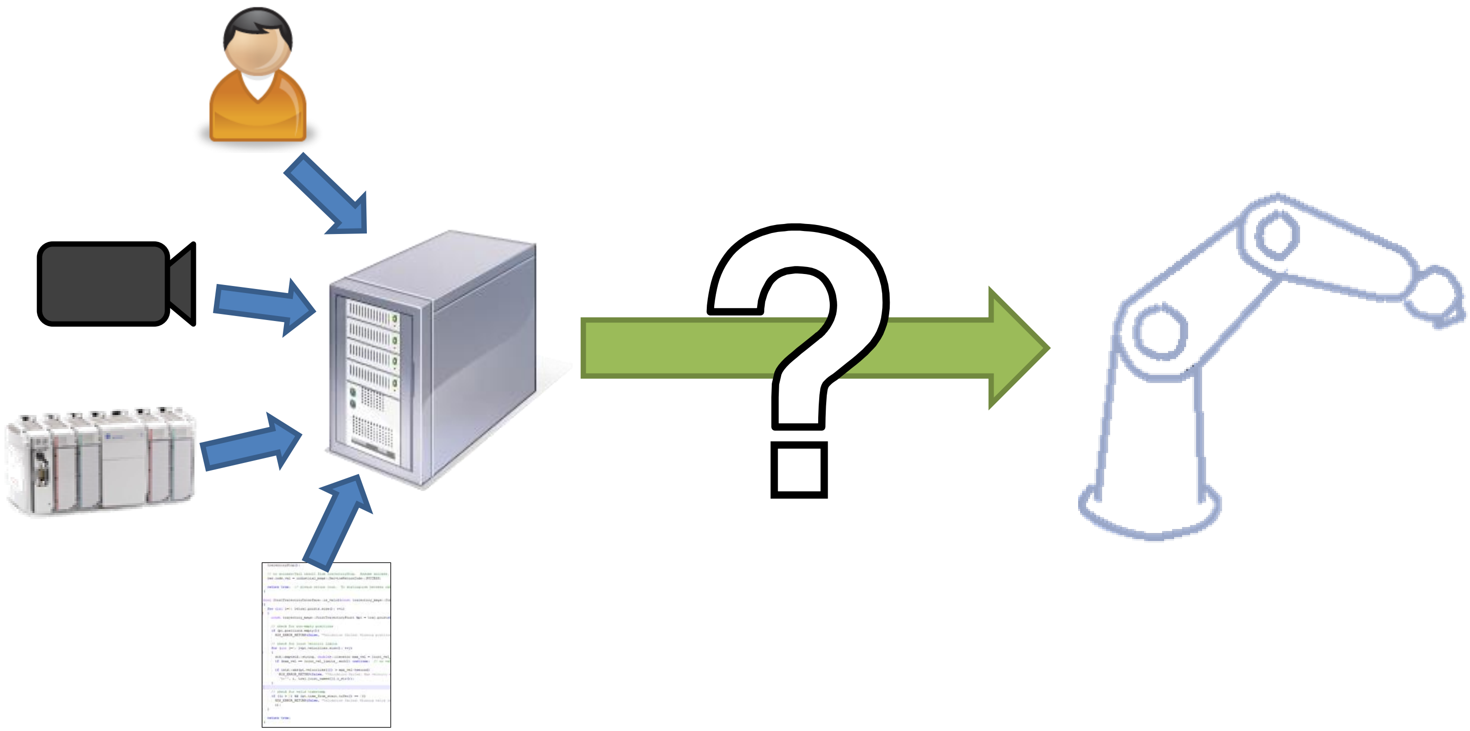
world->target = world->camera
* camera->target



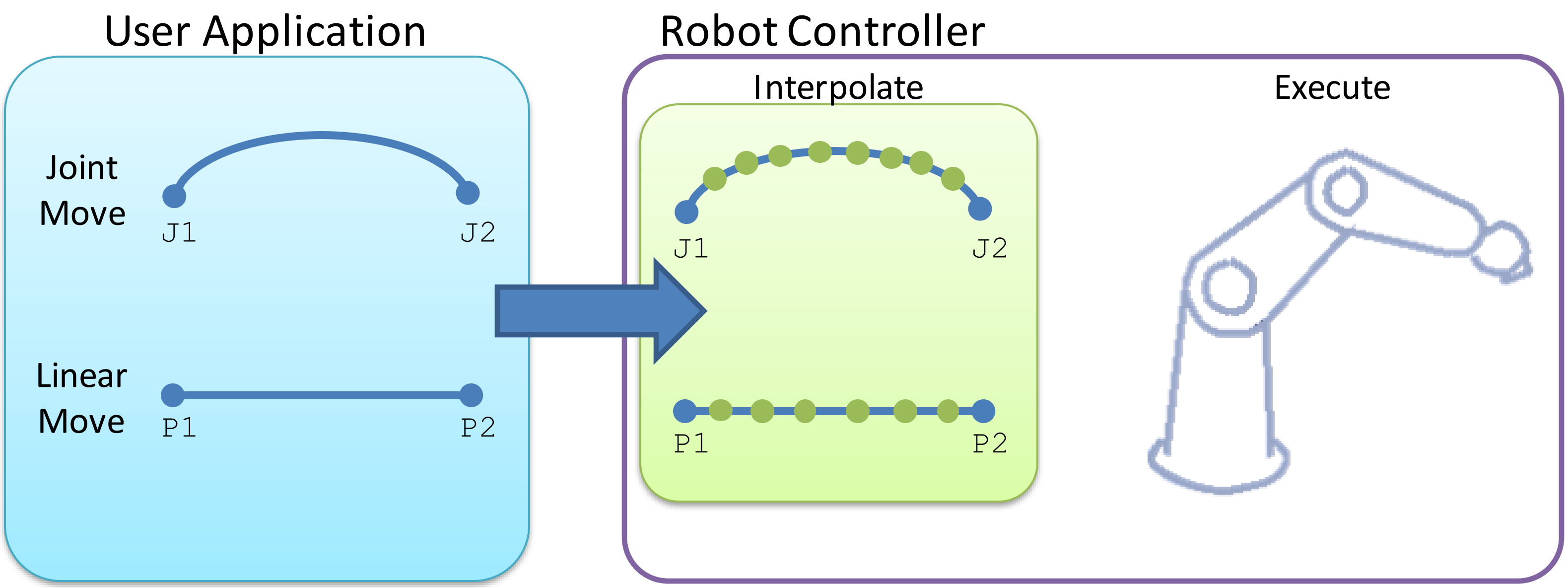
Motion Planning in ROS

(using MoveIt)

Motion Planning in ROS

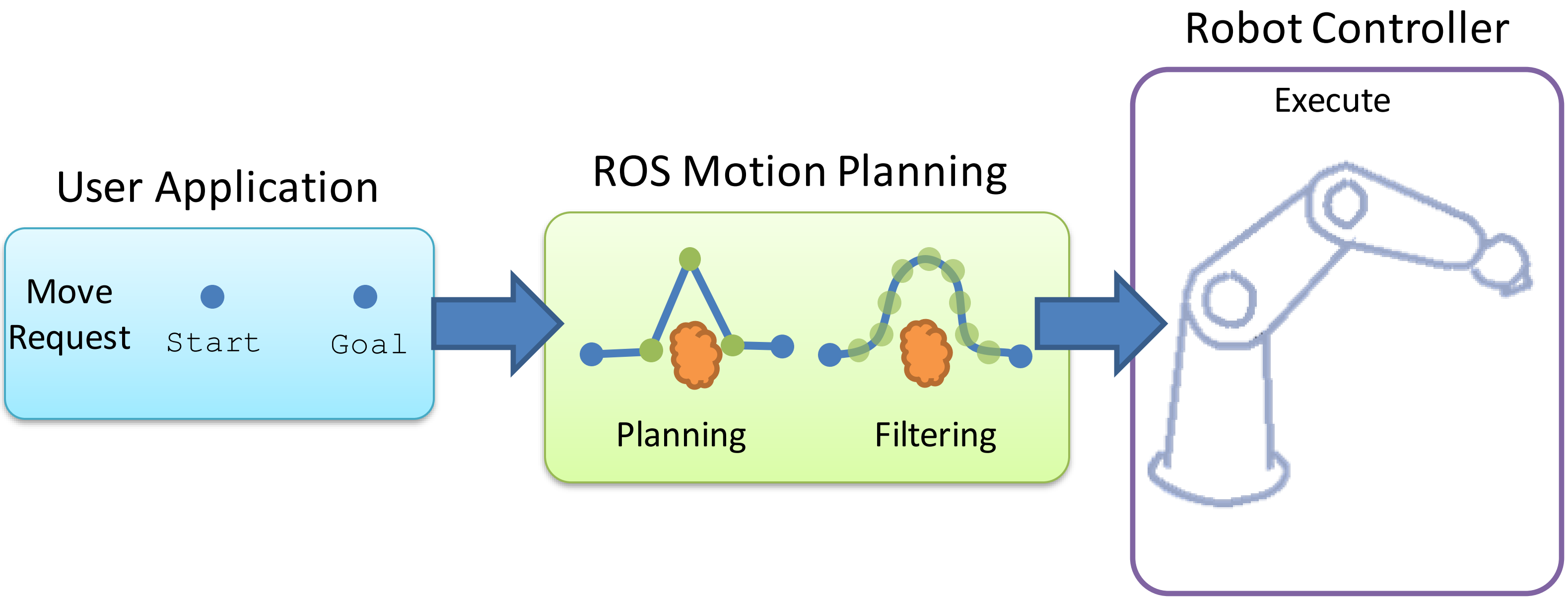


Traditional Robot Programming



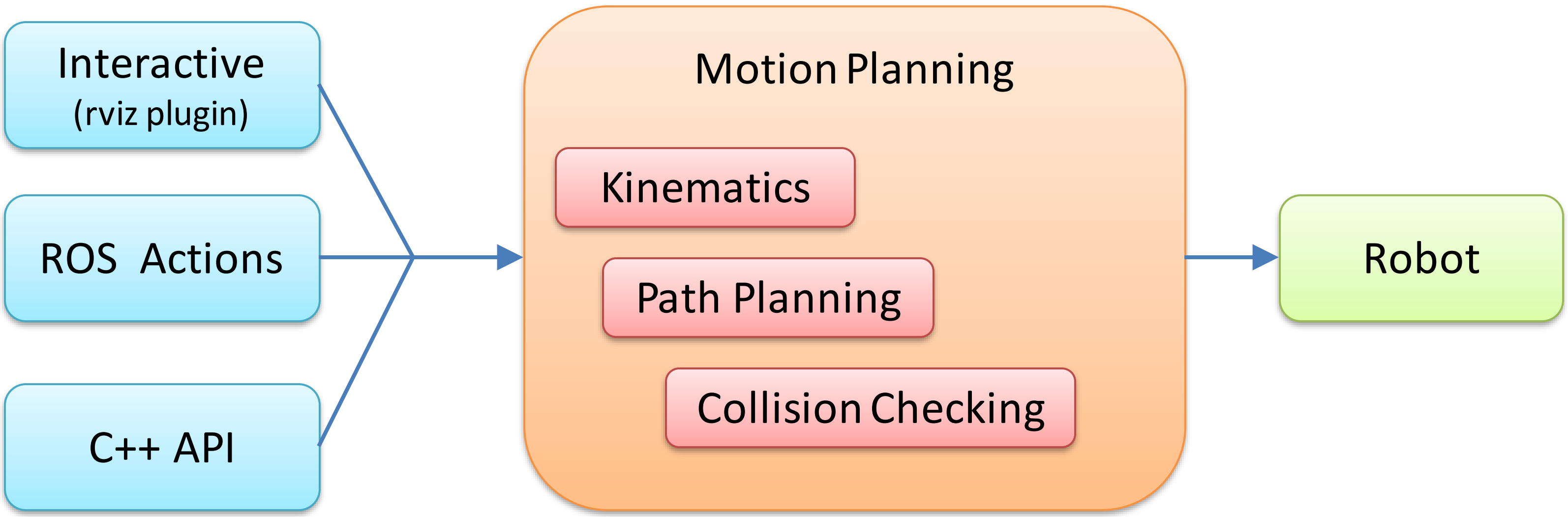
- Motion Types: *limited, but well-defined. One motion task.*
- Environment Model: *none*

ROS Motion Planning

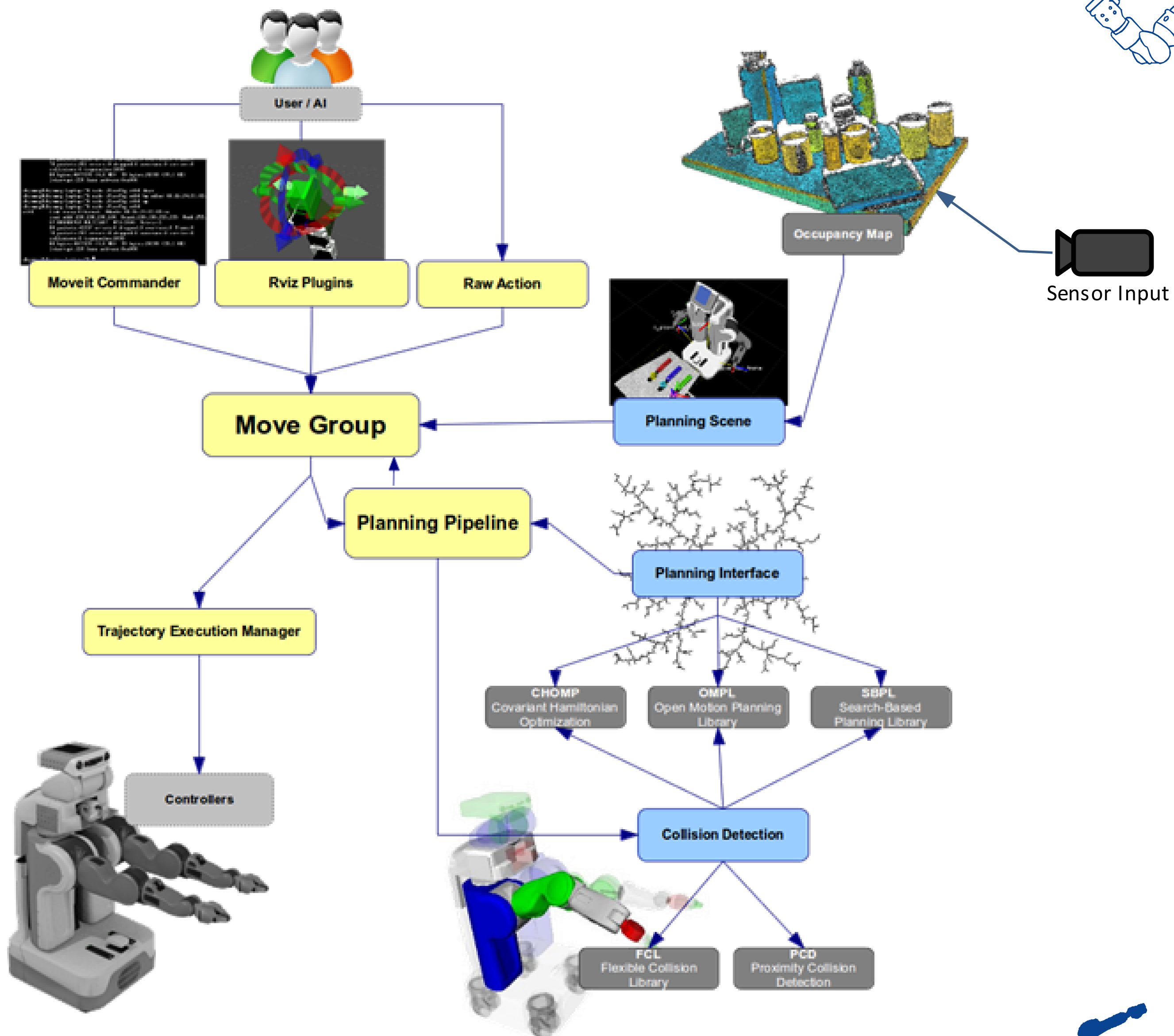


- Motion Types: *flexible, goal-driven, with constraints*
  but minimal control over actual path
- Environment Model: *yes (fixed CAD or sensor-driven)*

Motion Planning Components

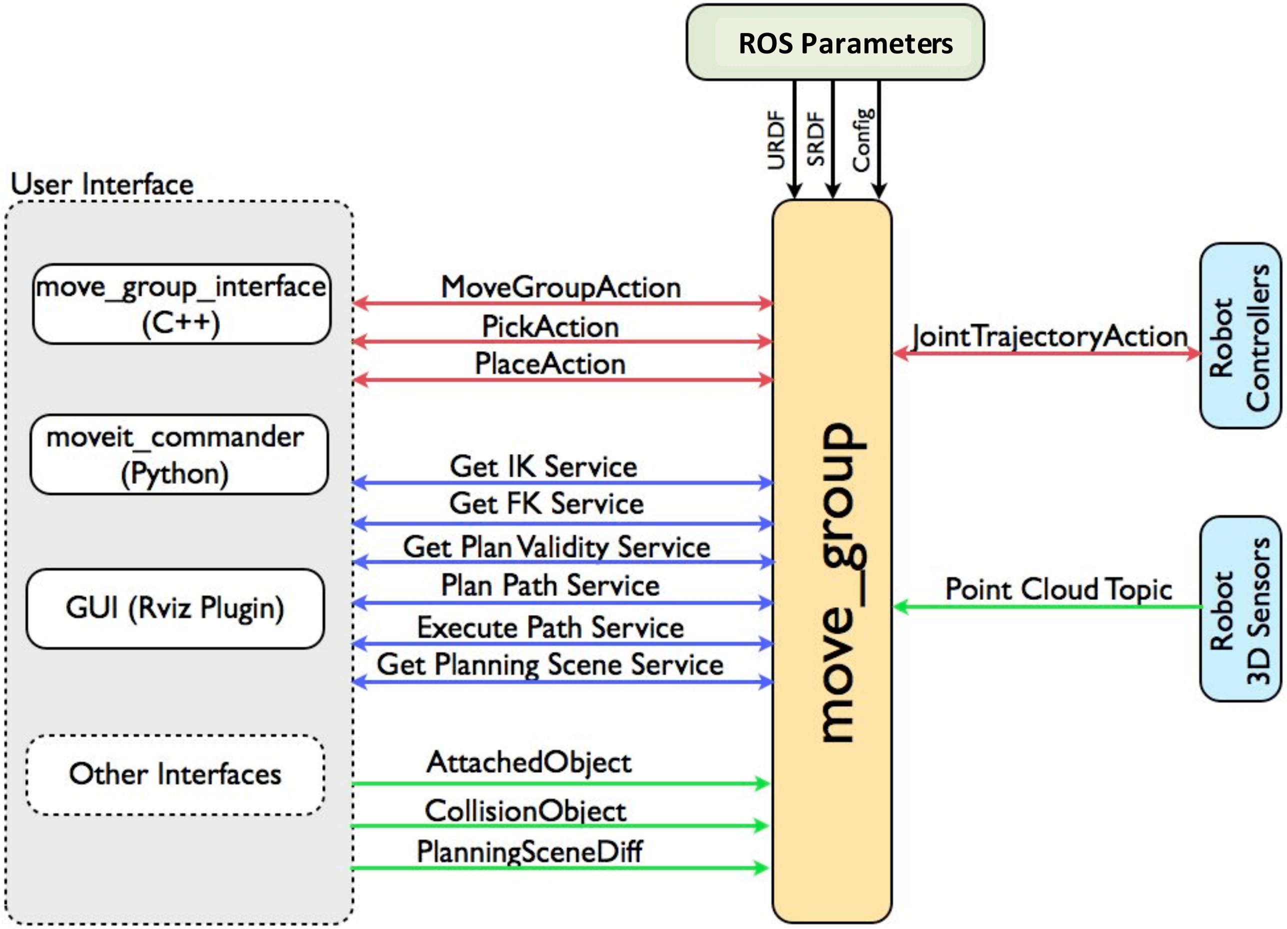


MoveIt Components



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/

MoveIt! Package - 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

MoveIt! Package Contents

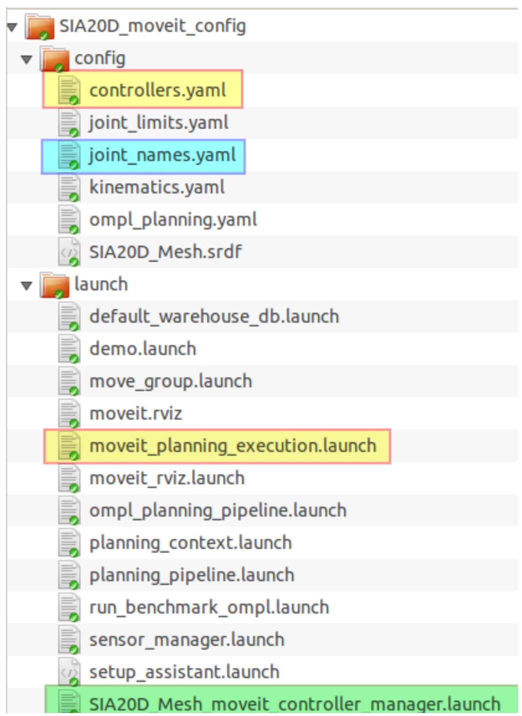
- A MoveIt! Package...
  - includes all required nodes, config, launch files
    - motion planning, filtering, collision detection, etc.
  - is unique to each individual robot model
    - includes references to URDF robot data
  - uses a standard interface to robots
    - publish trajectory, listen to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

Create a MoveIt! Package

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

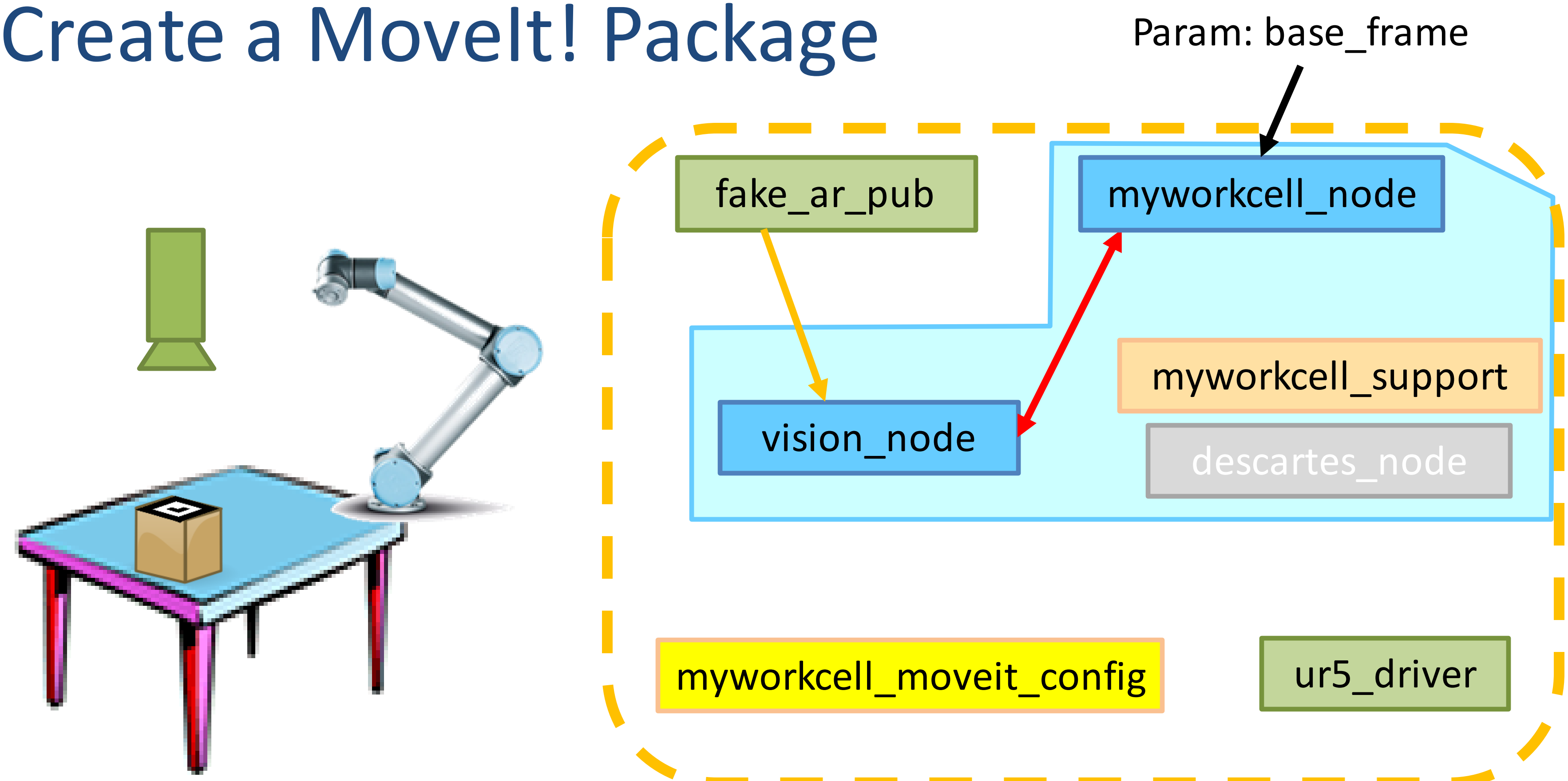
Update MoveIt! Package

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



Exercise 3.3:

Create a MoveIt! Package

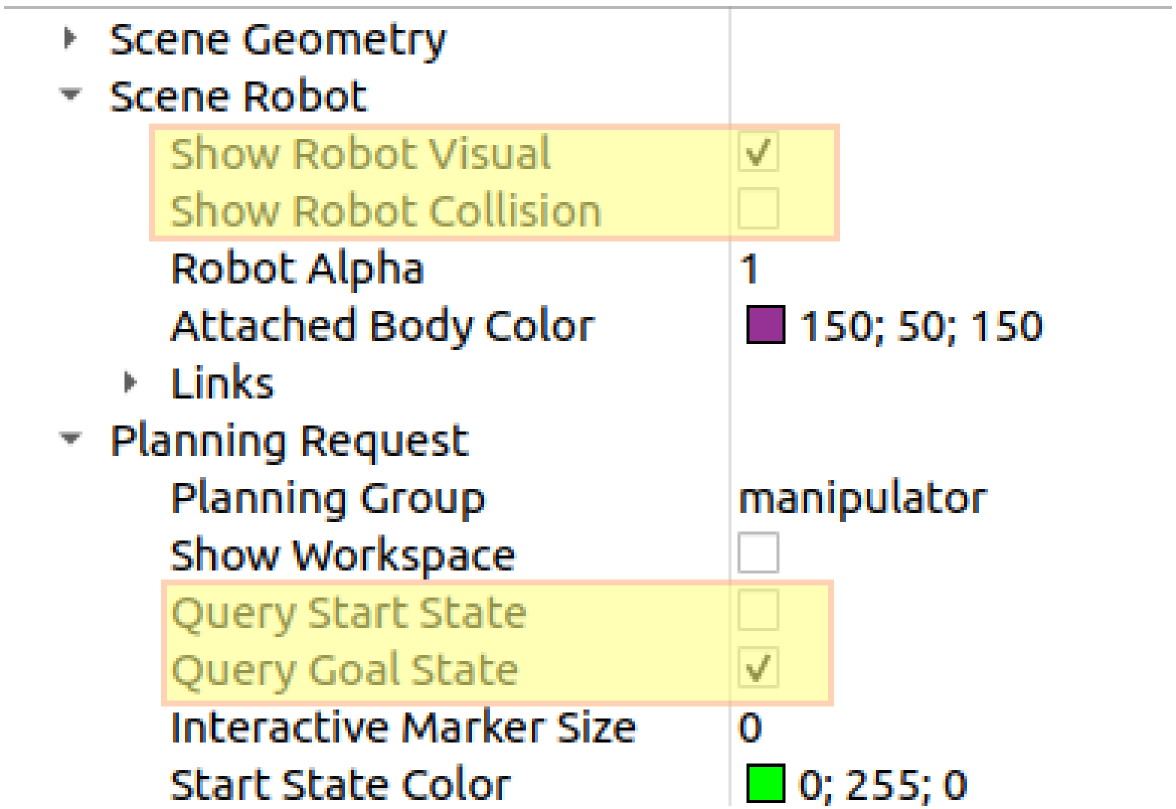


Motion Planning using MoveIt!

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

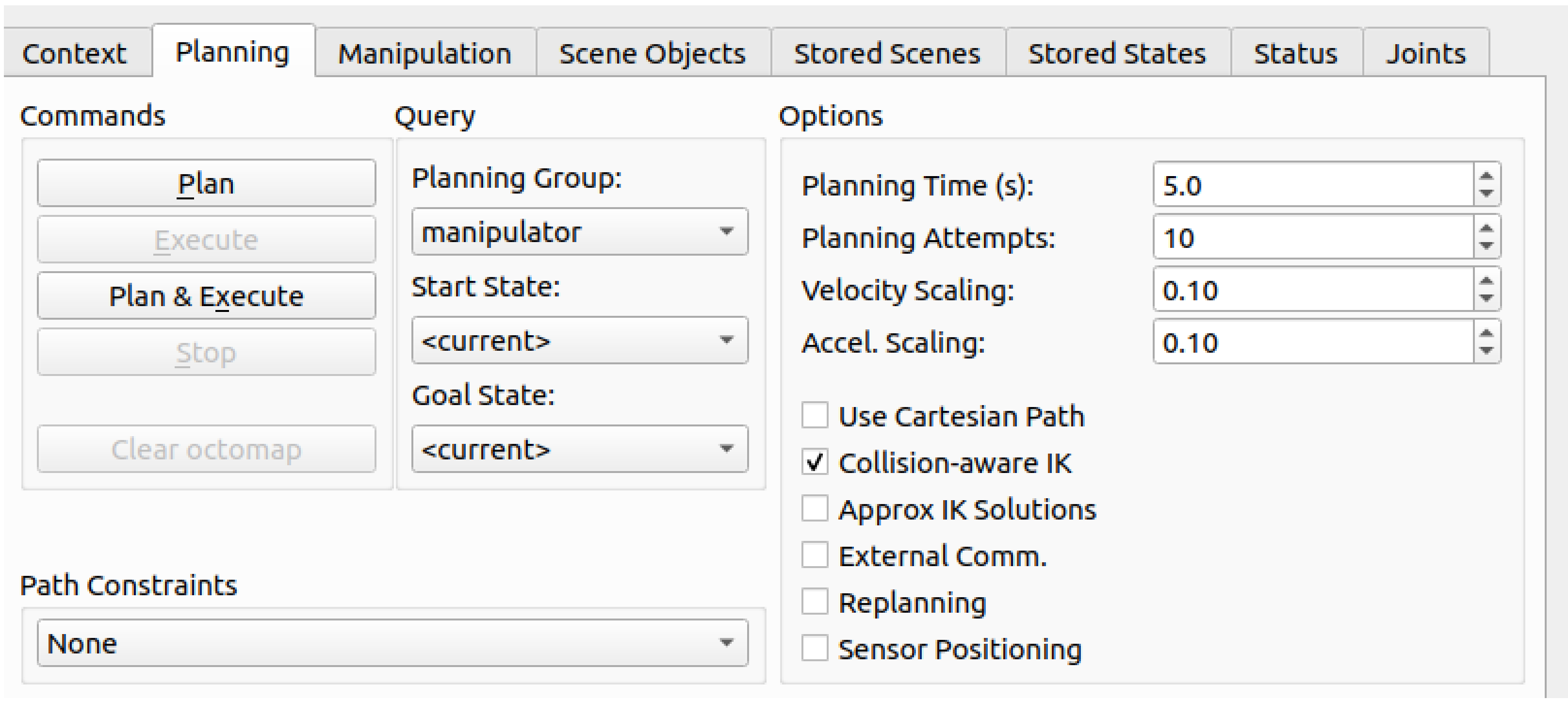
Motion Planning in RViz

Display Options



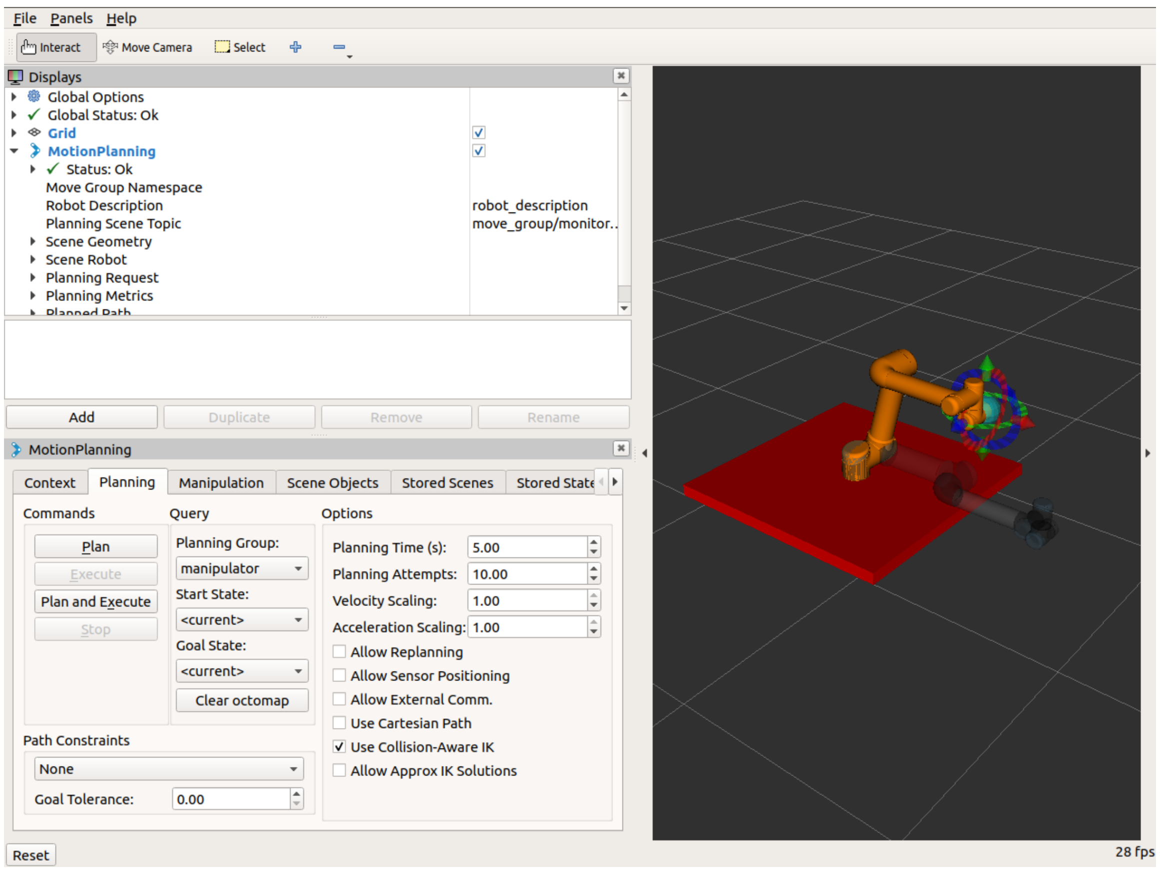
Motion Planning in RViz

Planning Options



Exercise 3.4:

Motion Planning using RVIZ



Review

ROS

- URDF
- MoveIt
- Path Planners
- RViz Planning

ROS-Industrial

- Robot Drivers
- Path Planners

Questions?

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