ROS-Industrial Basic Developer’s Training Class

Southwest Research Institute

Session 4:
Motion Planning

Moveit! Planning using C++

Descartes

Intro to Perception

Southwest Research Institute

Common Motion Planners

Motion Planner	Application Space	Notes
Descartes	Cartesian path planning	Globally optimum; sampling-based search; Captures “tolerances”
CLIK	Cartesian path planning	Local optimization; Scales well with high DOF; Captures “tolerances”
TrajOpt	Free-space Planning	Optimizes by following cost gradients; Fast! Requires convex models
STOMP	Free-space Planning	Optimizes by random perturbations; Emphasizes smooth paths
OMPL / MoveIt	Free-space Planning	Stochastic sampling; Easy and convenient interface

Motion Planning in C++

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

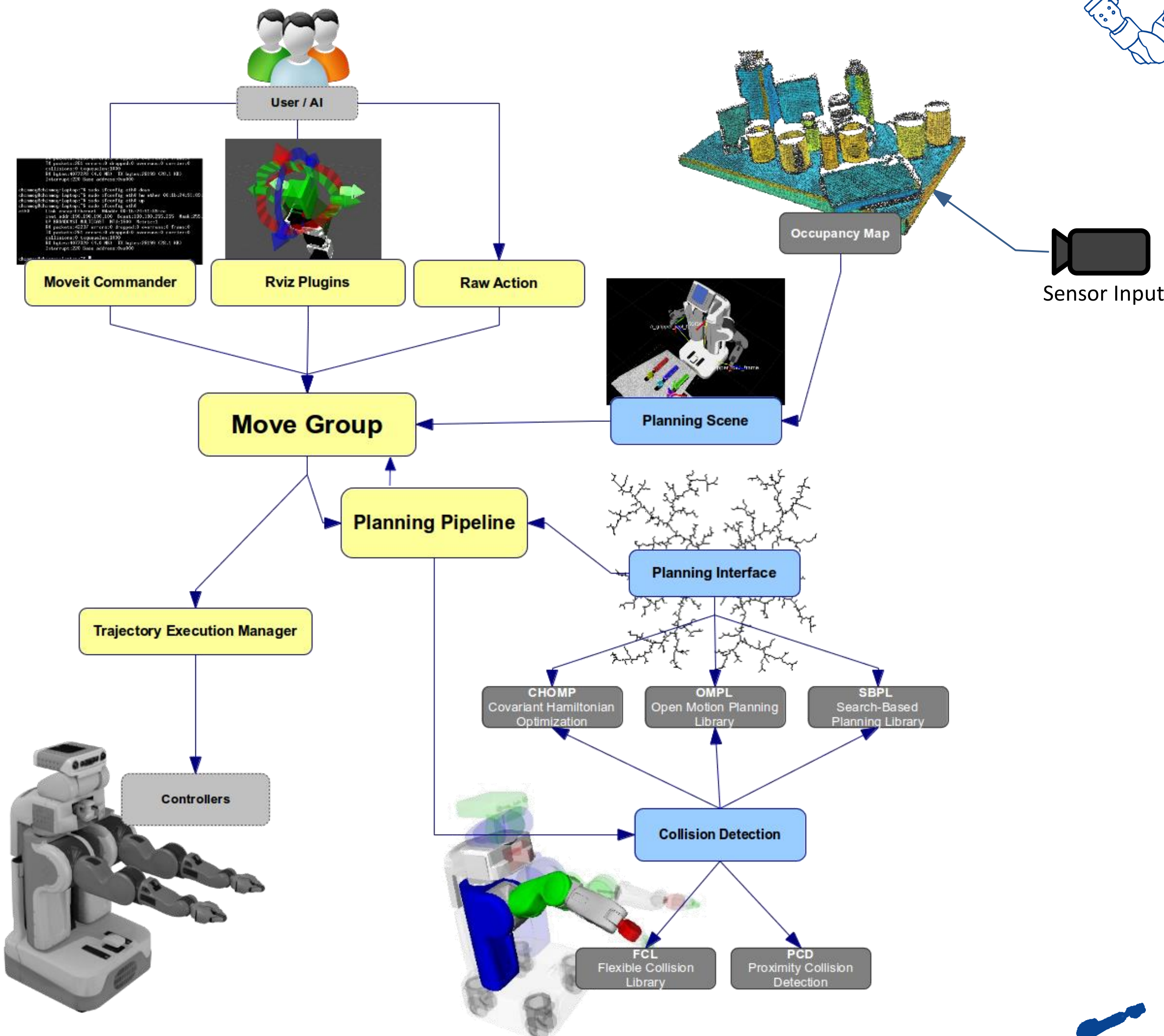
move_group_interface

```
#include <moveit/move_group_interface/move_group_interface.h>
...
Moveit::planning_interface::MoveGroupInterface group("manipulator");
group.setRandomTarget();
group.move();
```

3 lines = collision-aware path planning & execution

Amazing!

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();
```

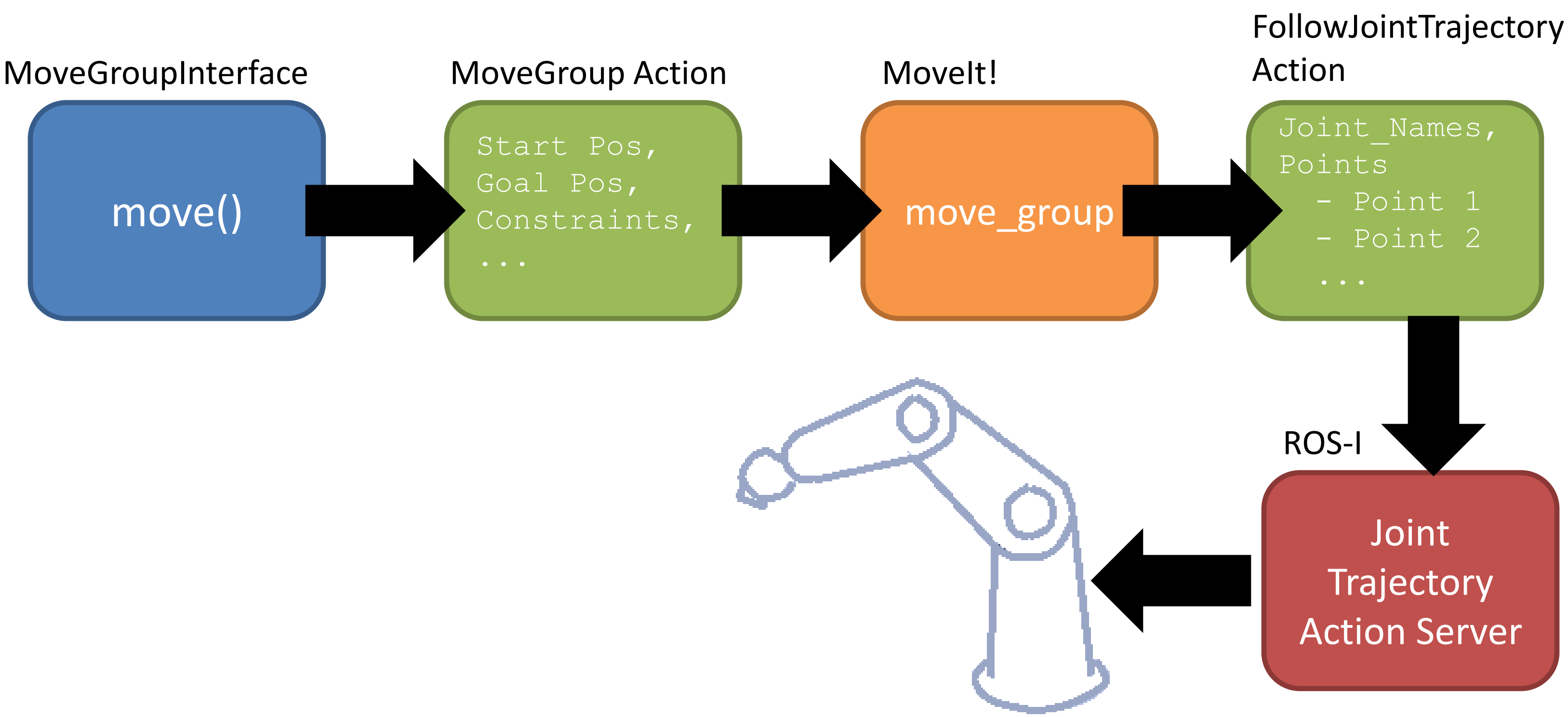
Joint position:

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

Cartesian position:

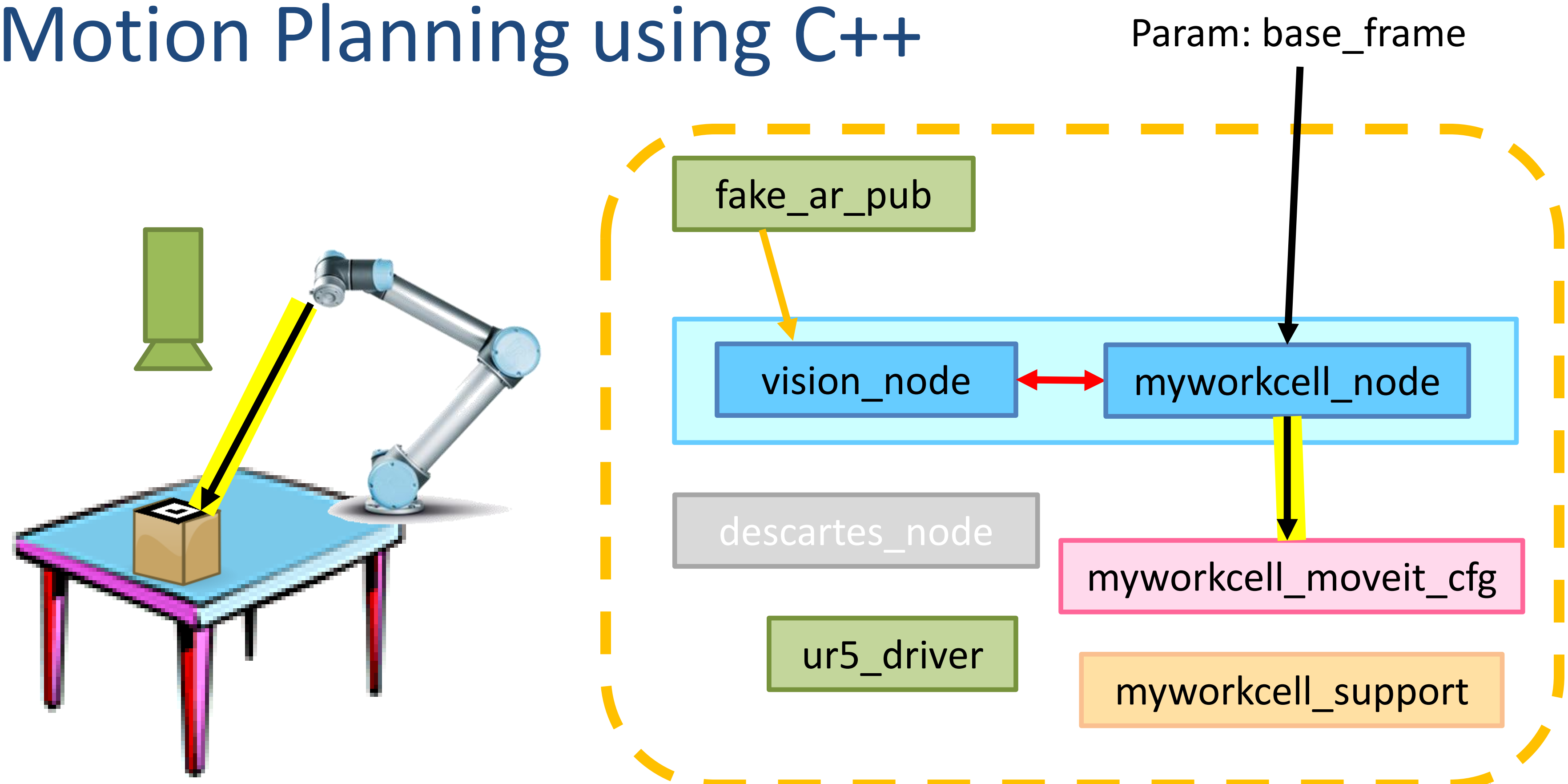
```
Affine3d pose = my_function();
group.setPoseTarget(pose);
group.move();
```

Behind the Scenes



Exercise 4.0:

Motion Planning using C++



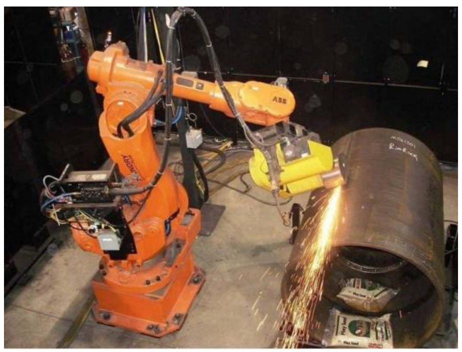
INTRODUCTION TO DESCARTES

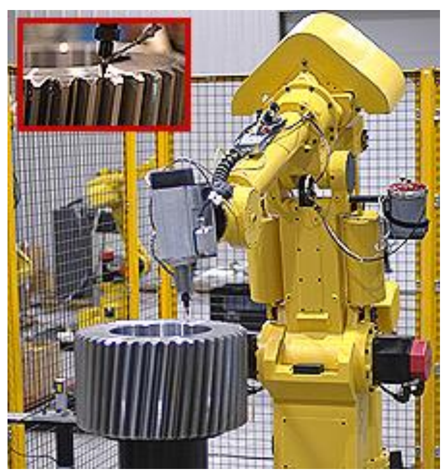
Outline

- Introduction
- Overview
  - Descartes architecture
- Path Planning
  - Exercise 4.1

Introduction

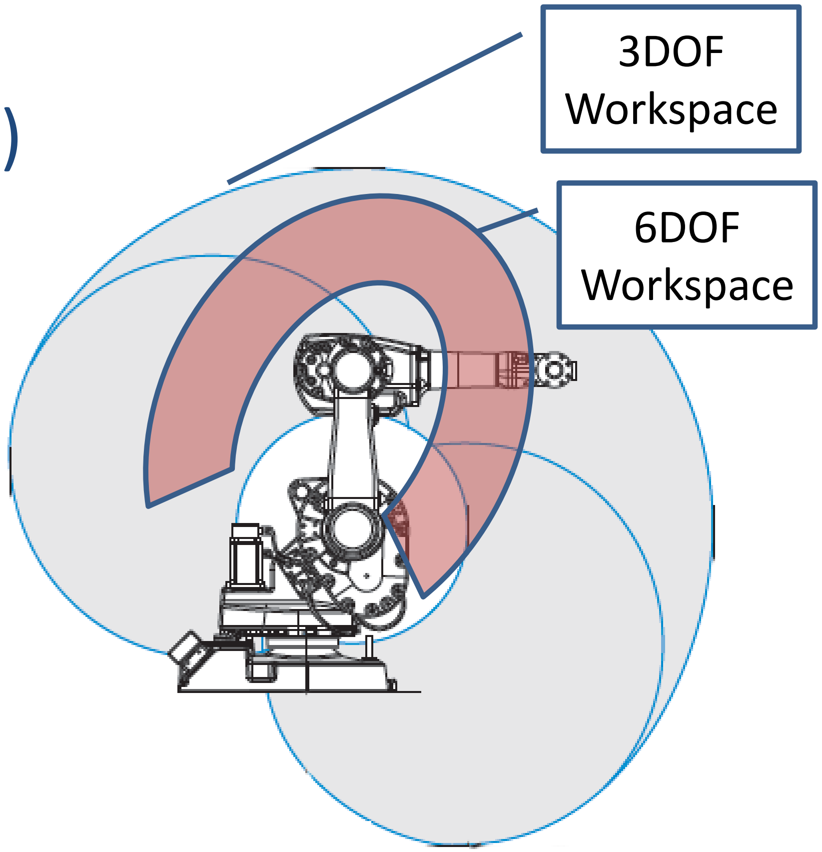
- Application Need:
  – Semi-constrained trajectories: traj. DOF < robot DOF





Current Solution

- Fully-specified Paths
  - 6DOF poses (even if arbitrary)
  - Redundant axes known
  - ➢ Joint trajectory is set by IK



- Limitations
  - Reduced workspace
  - Relies on human intuition
  - Collisions, singularities, joint limits

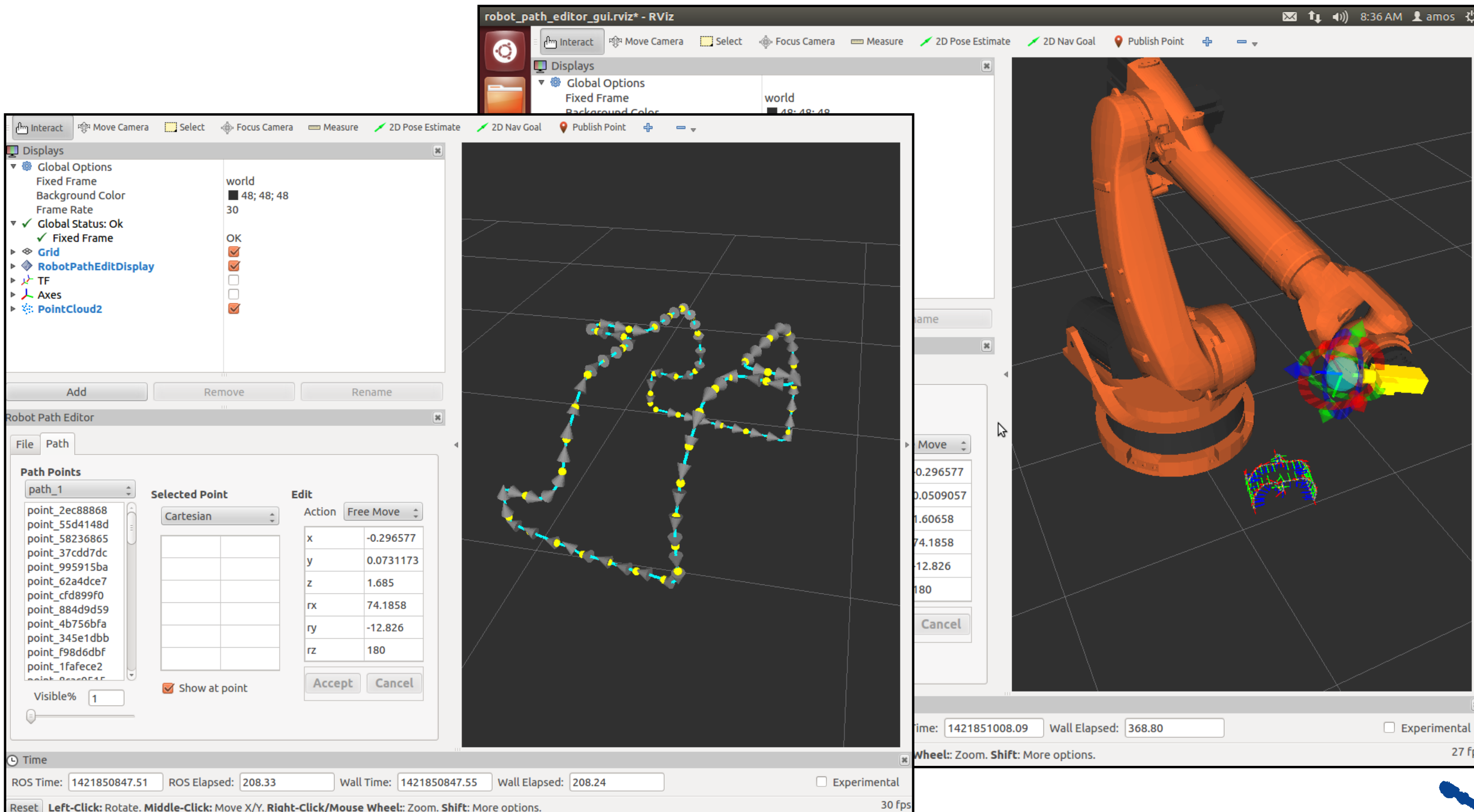
Descartes

- Planning library for semi-constrained trajectories

- Requirements
  - Generate well-behaved plans that minimize joint motions
  - Handle *hybrid* trajectories
    - joint, Cartesian, specialized points

Descartes Use Case

- Robotic Routing



Other Uses

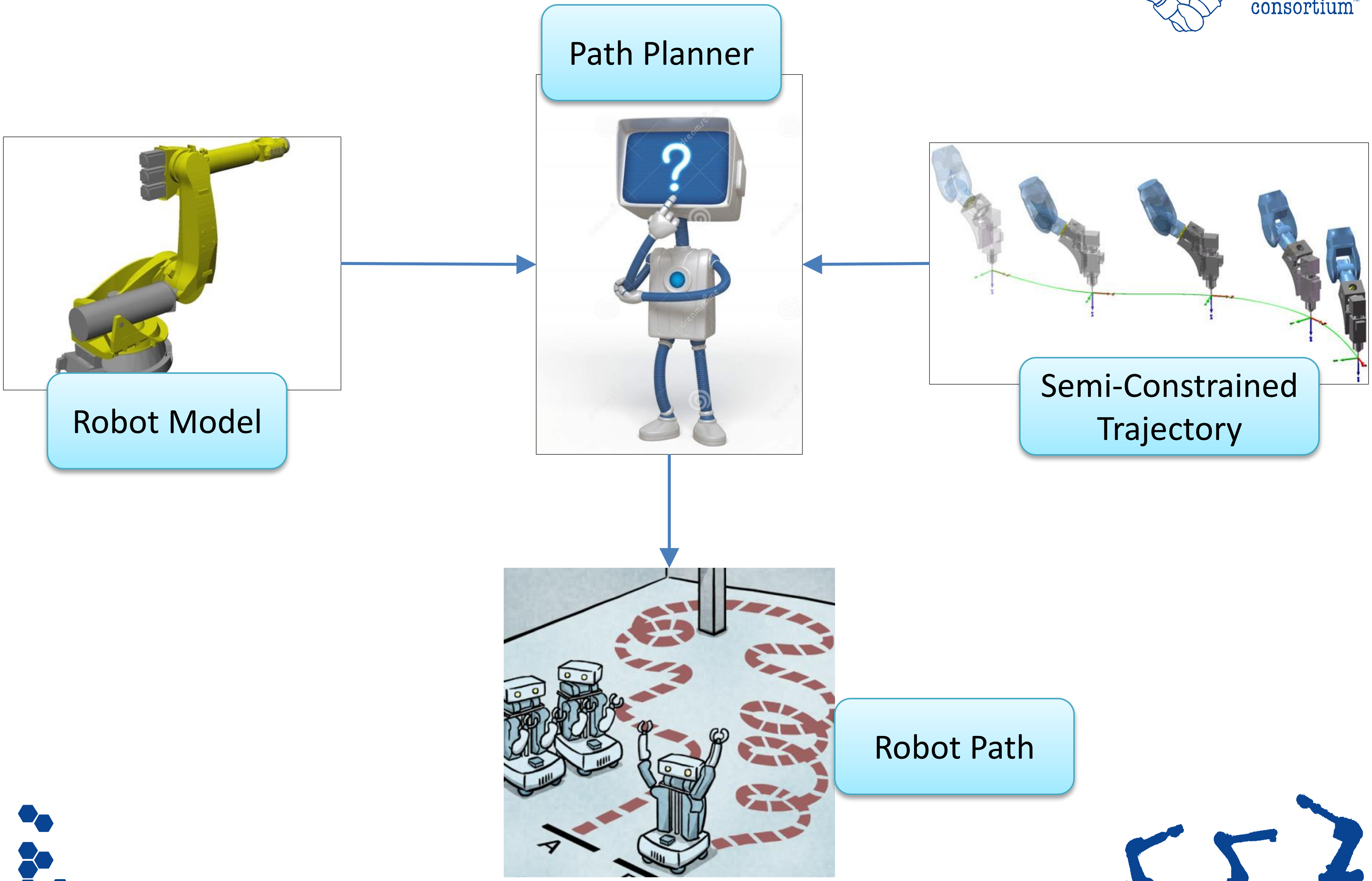
- Robotic Blending

https://www.youtube.com/watch?v=cMZaxKsQdhg

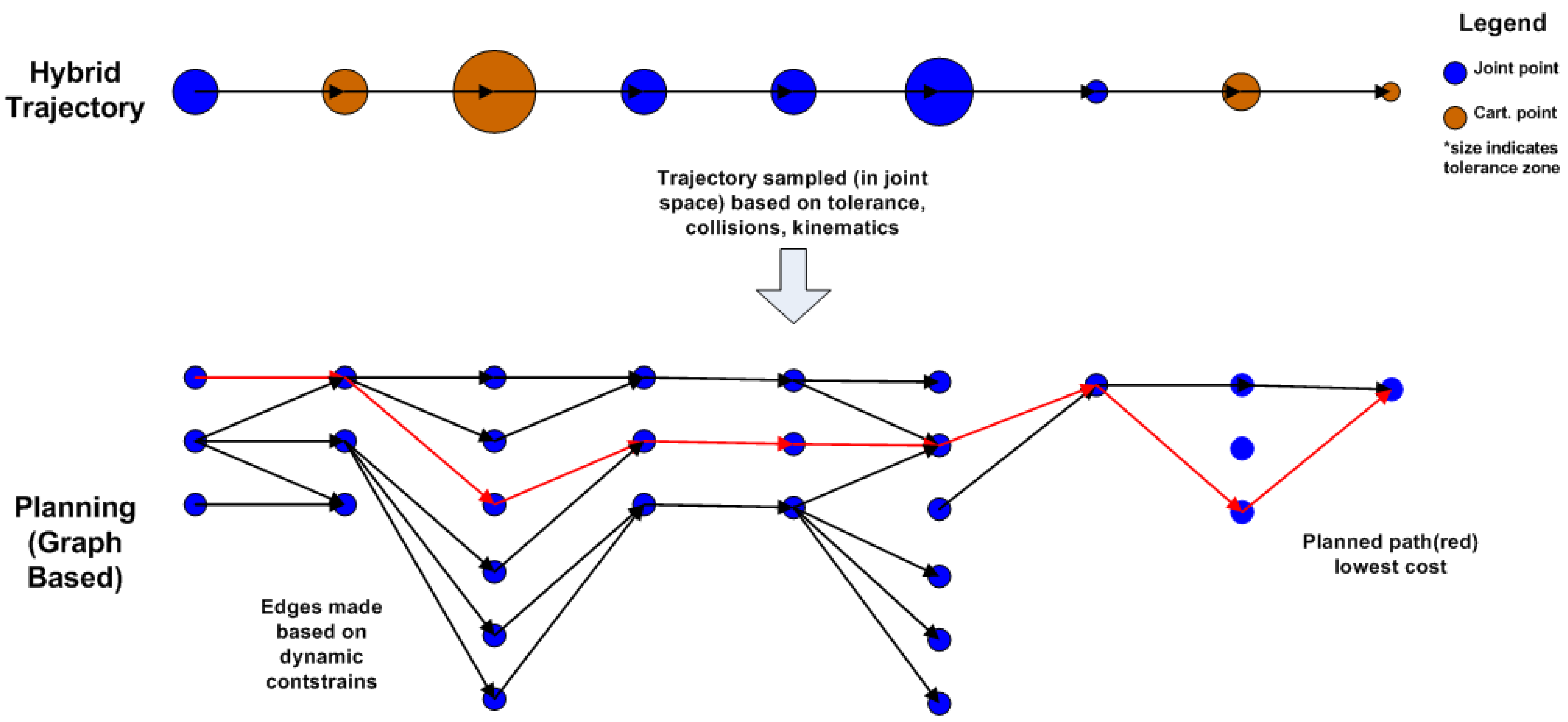
Open Source Details

- Public development
  - https://github.com/ros-industrial-consortium/descartes
- Wiki Page
  - http://wiki.ros.org/descartes
- Acknowledgements:
  - Dev team: Dan Solomon (former SwRI), Shaun Edwards (former SwRI), Jorge Nicho (SwRI), Jonathan Meyer (SwRI), Purser Sturgeon(SwRI)
  - Supported by: NIST (70NANB14H226), ROS-Industrial Consortium FTP

Descartes Workflow



Descartes Approach



Descartes Interfaces

- Trajectory Points
  - Cartesian point
  - Joint point
  - AxialSymmetric point (5DOF)
- Robot Model
  - MoveIt wrapper (working with MoveIt to make better)
  - FastIK wrappers
  - Custom solution
- Planners
  - Dense – graph based search
  - Sparse – hybrid graph based/interpolated search

Descartes Interfaces

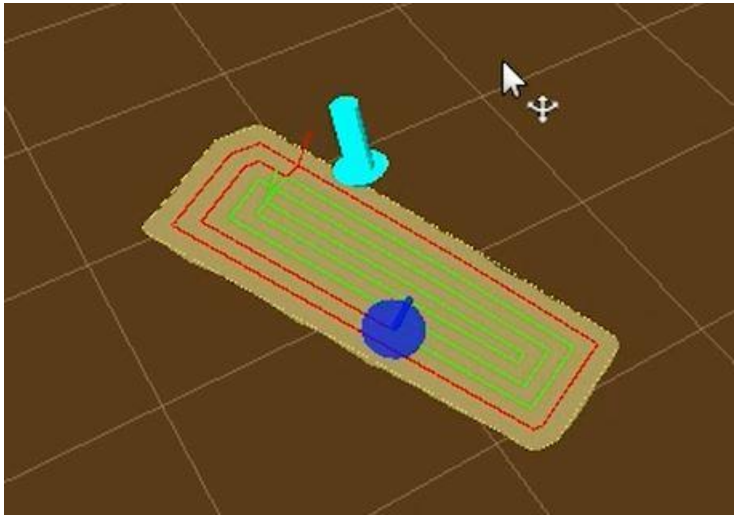
- Trajectory Points
  - JointTrajectoryPt
    - Represents a robot joint pose. It can accept tolerances for each joint
  - CartTrajectoryPt
    - Defines the position and orientation of the tool relative to a world coordinate frame. It can also apply tolerances for the relevant variables that determine the tool pose.
  - AxialSymmetricPt
    - Extends the CartTrajectoryPt by specifying a free axis of rotation for the tool. Useful whenever the orientation about the tool's approach vector doesn't have to be defined.

Descartes Interfaces

- Planners
  - Planners are the highest level component of the Descartes architecture.
  - Take a trajectory of points and return a valid path expressed in joint positions for each point in the tool path.
  - Two implementations
    - DensePlanner
    - SparsePlanner

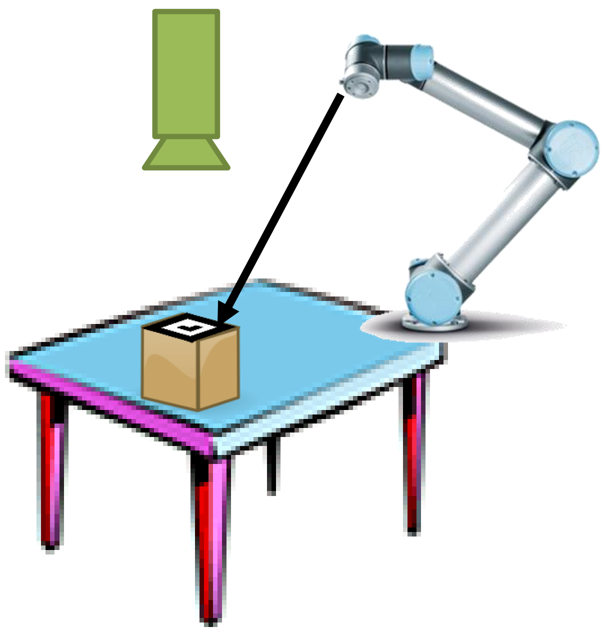
Descartes Input/Output

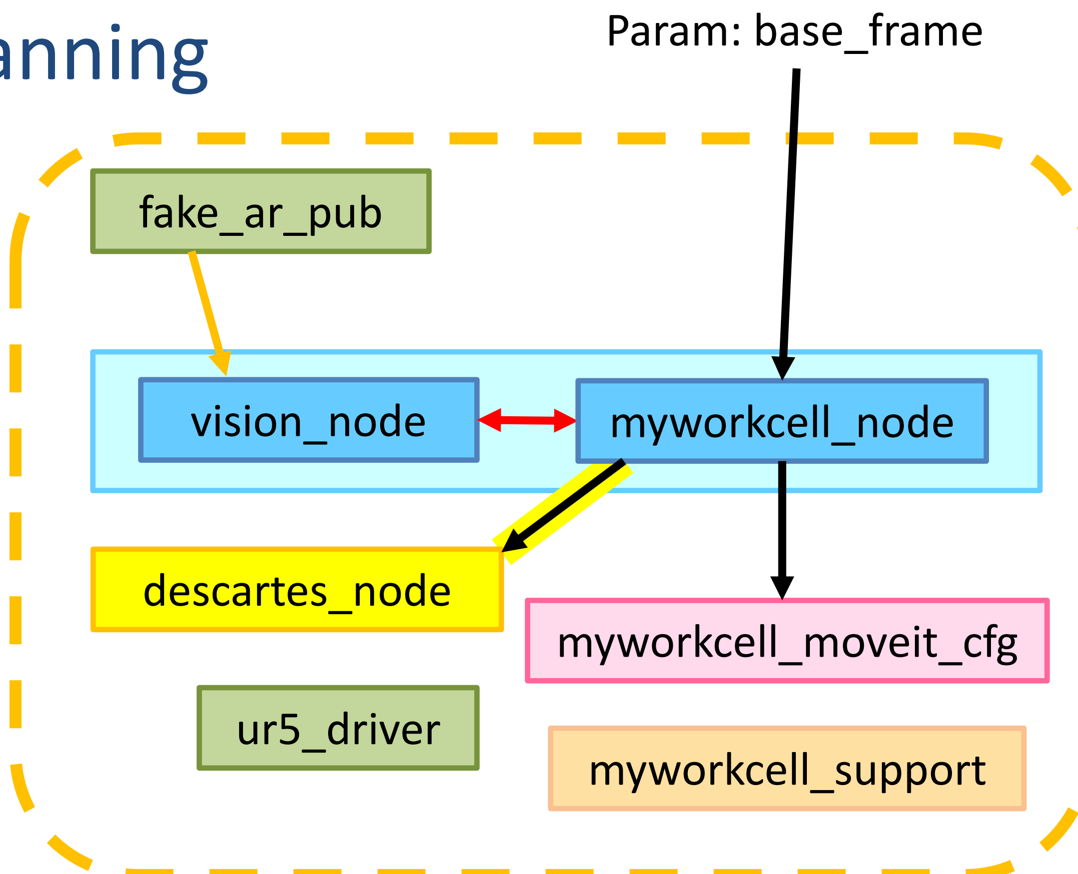
- Input:
  - Can come from CAD
  - From processed scan data
  - Elsewhere
- Output
  - Joint trajectories
  - Must convert to ROS format to work with other ROS components (see 4.0)



Exercise 4.1:

Descartes Path Planning





INTRODUCTION TO PERCEPTION

Objectives

- Brief overview of the perception capabilities available in ROS
- Small exercise in 3D data, PCL, and Rviz

- Re-visited during Advanced Training

Why Perception?

- ROS applications frequently deal with **dynamic** environments.
- Perception helps the robot understand how it needs to **adapt** its behaviors.
- Perception needs are often very application-specific, but ROS provides a wide variety of **enabling capabilities**.

2D Camera Drivers

- See “Cameras” overview page:
  - http://wiki.ros.org/Sensors/Cameras
  - 20+ drivers
    - Protocols: USB, Firewire, Ethernet, OpenCV, ...
    - Manufacturers: Basler, Prosilica, Canon, ...

- Standardized Messages:
  - sensor_msgs/Image
  - sensor_msgs/CameraInfo

3D Camera Drivers

- ROS-I 3D Camera Survey
  - https://rosindustrial.org/3d-camera-survey/
- Structured Light Cameras
  - Kinect1/XtionPro: openni_launch, openni2_launch

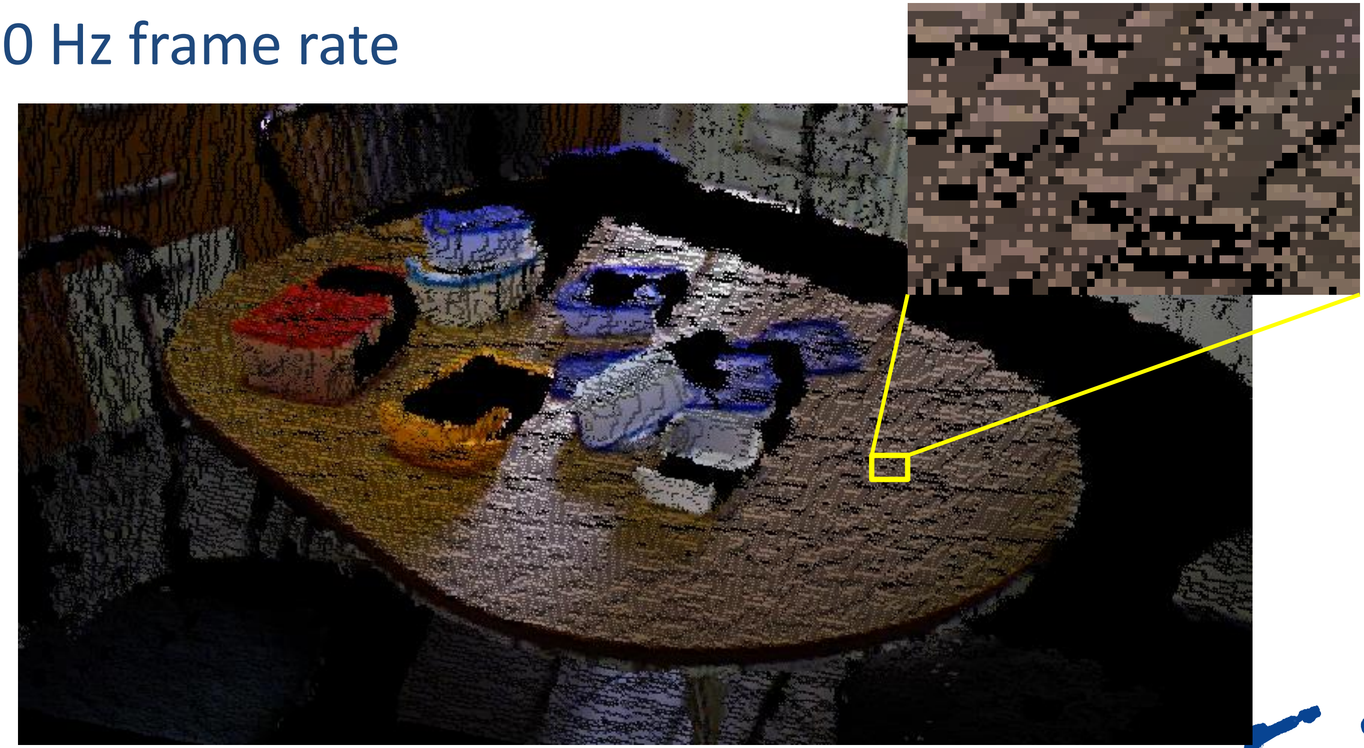
- TOF Cameras
  - Kinect2: iai_kinect2
  - IFM Efector: o3d3xx_ros

- Stereo Vision
  - Intel RealSense: realsense_camera
  - Carnegie Robotics MultiSense: multisense_ros

- 2D/3D laser scanners

3D Cameras

- Produce (colored) point cloud data
- Huge data volume
  - Over 300,000 points per cloud
  - 30 Hz frame rate



Perception Processing Pipeline

- Goal: Gain knowledge from sensor data

Objective	Processing
Improve data quality	Filters, outlier rejection
Speed up processing	Downsampling
Unified world view	Merge multiple data sources
Build model from data	Segmenting, Model fitting
Gain knowledge	Classify, measure

2D Processing (OpenCV)

- Open Computer Vision (OpenCv) - http://opencv.org/ *(not ROS-specific)*
  - Filter / Threshold
  - Calibration
  - Feature/Object Detection
  - Video (motion, foreground)

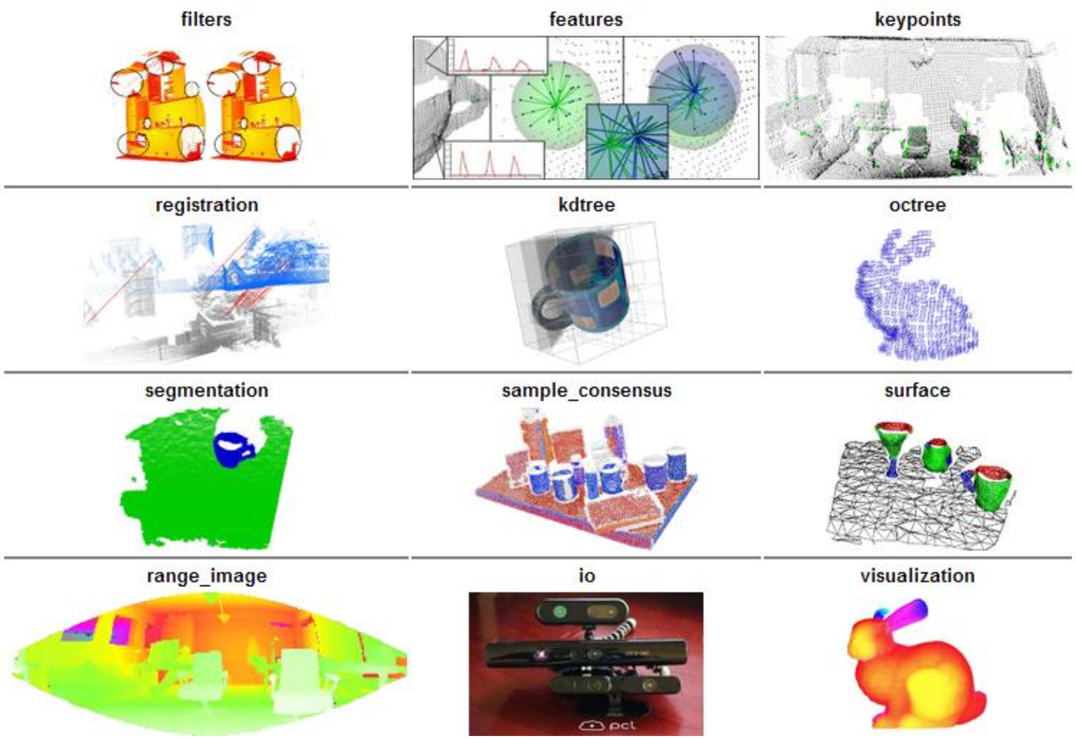
http://opencv.org

OpenCV in ROS

- cv_bridge
  - Converts ROS msg (sensor_msgs/Image) to OpenCV Image
- opencv3
  - ROS packaging of OpenCV library
- vision_opencv/Tutorials

3D Processing (PCL)

- Point Cloud Library (PCL) - http://pointclouds.org/ *(not ROS-specific)*
  - Focused on 3D Range(Colorized) data



http://pointclouds.org

PCL in ROS

- pcl_ros
  - Converts ROS msg (sensor_msgs/PointCloud2) to PCL PointCloud
- pcl/Tutorials
- PCL Tools
  - apt install pcl-tools
  - 140+ command-line tools, wrapping PCL functions

Industrial Calibration

- General-purpose calibration library
  - based on Google's Ceres non-linear optimization solver
  - can solve a wide range of calibration scenarios
    - robot to workspace
    - multi-camera alignment
    - camera mounting: fixed (workcell) vs. robot-mounted

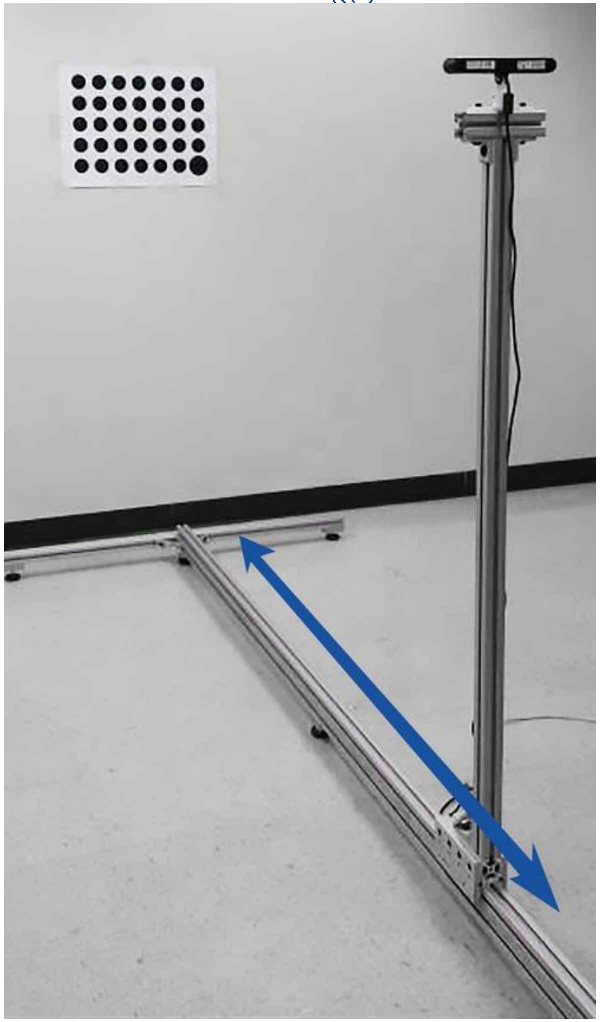
- For more information:
  - Library: Wiki , GitHub
  - Tutorials: GitHub

Intrinsic Calibration

- **Goal**: Calculate optical characteristics of a camera + lens
  - Focal Length, optical center, skew

- **It's Optional**... can use default parameters from mfg.
  - Camera-specific calibration improves accuracy

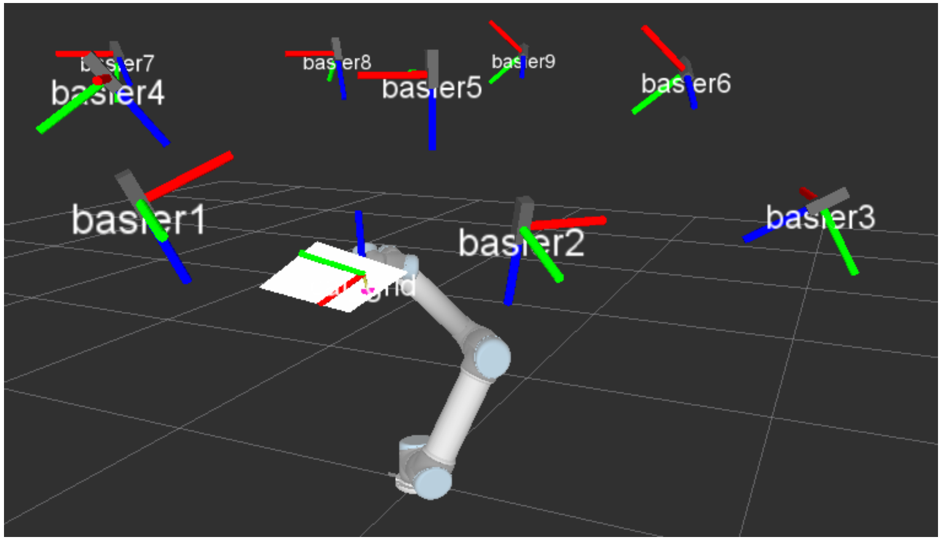
- **In Development**: Automatic Intrinsic Calibration using known movements of a linear rail.



http://wiki.ros.org/industrial_extrinsic_cal/Tutorials/Intrinsic%20Camera%20Calibration

Extrinsic Calibration

- **Goal**: Calculate camera position relative to world (or robot)
- Required for image processing in “real world” units



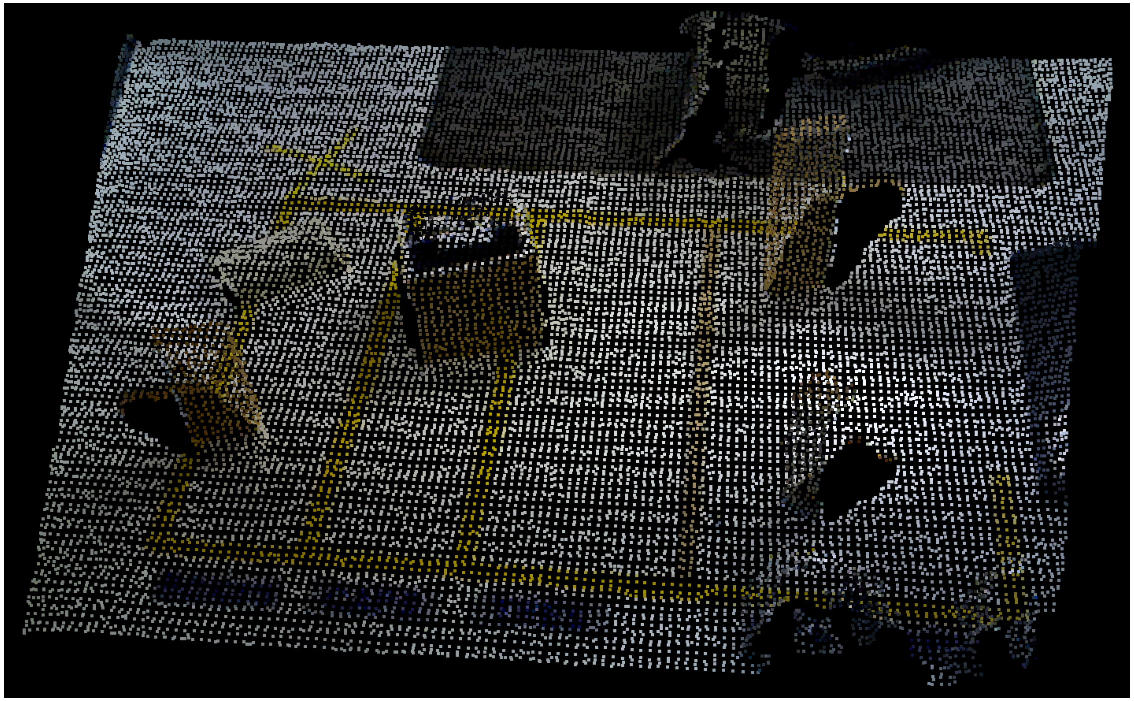
https://www.youtube.com/watch?v=MJFtEr_Y4ak

Many More Libraries

- Many more libraries in the ROS Ecosystem
  - ros-perception - Perception meta-repository
  - AR Tracker (ar_track_alvar)
  - Robot Self Filter (robot_self_filter)
  - Laser-Scan Filters (laser_filters)
  - Laser-Scan to Point Cloud (laser_assembler)

Exercise 4.2

- Play with PointCloud data
  - Simulate sensor data (RGBD image)
  - Use PCL Command Line Tools



Review/Q&A

Session 3

Motion Planning

- URDF / Xacro
- TF
- MoveIt! Setup Assistant
- Motion Planning (Rviz)

Session 4

Motion Planning (C++)

Descartes

- Graph Search
- Trajectory points

Perception

- Sensor Drivers
- OpenCV (2D)
- PCL (3D)
- Calibration