ROS-Industrial Basic Developer's Training Class

October 2021

Southwest Research Institute

Session 4:
Motion Planning

Moveit! Planning using C++

Descartes

Intro to Perception

Southwest Research Institute

Motion Planning in C++

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

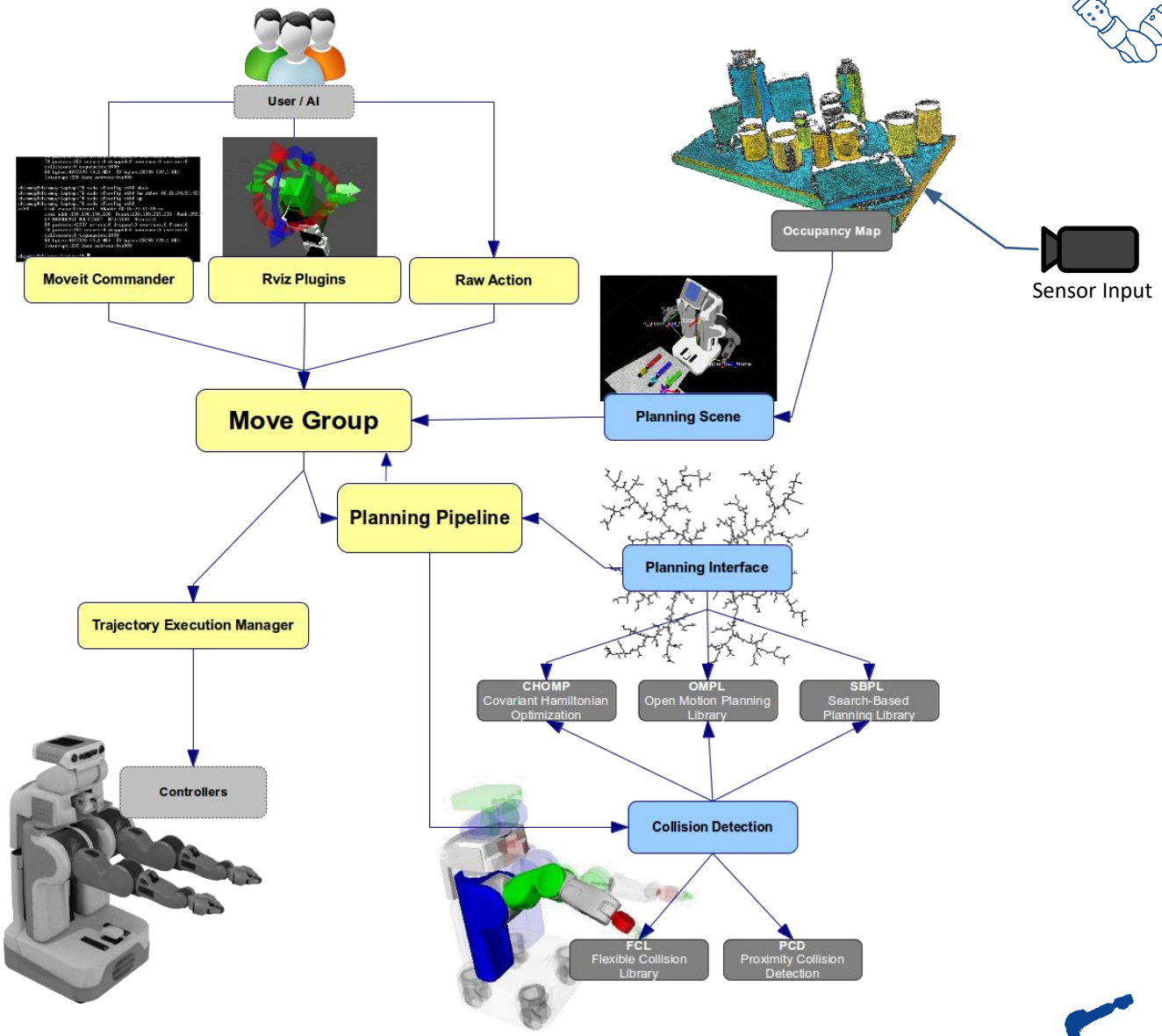
moveit_cpp

```
#include <moveit/moveit_cpp/moveit_cpp.h>
...
moveit_cpp::MoveItCpp::Ptr moveItCpp = make_shared(node);
moveit_cpp::PlanningComponent::Ptr planner = make_shared("arm", moveItCpp);

planner->setGoal("home");
planner->plan();
planner->execute();
```

5 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:

```
planner.setGoal("home");
```

Joint position:

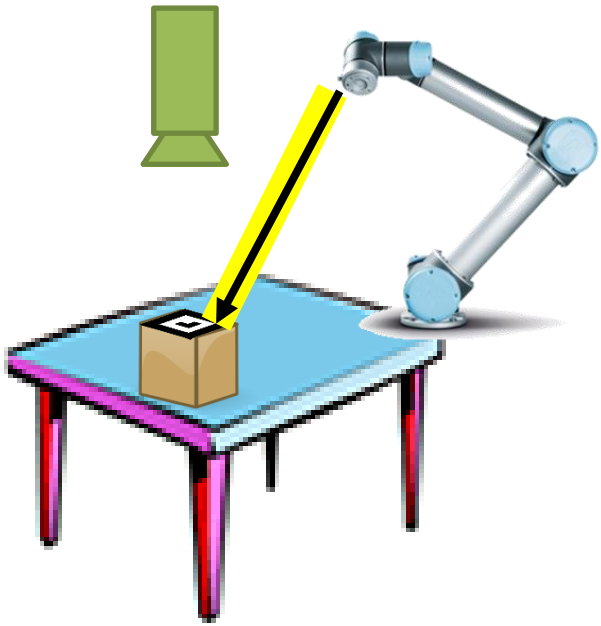
```
robot_state::RobotState joints.setStateValues(names, positions);
planner.setGoal(joints);
```

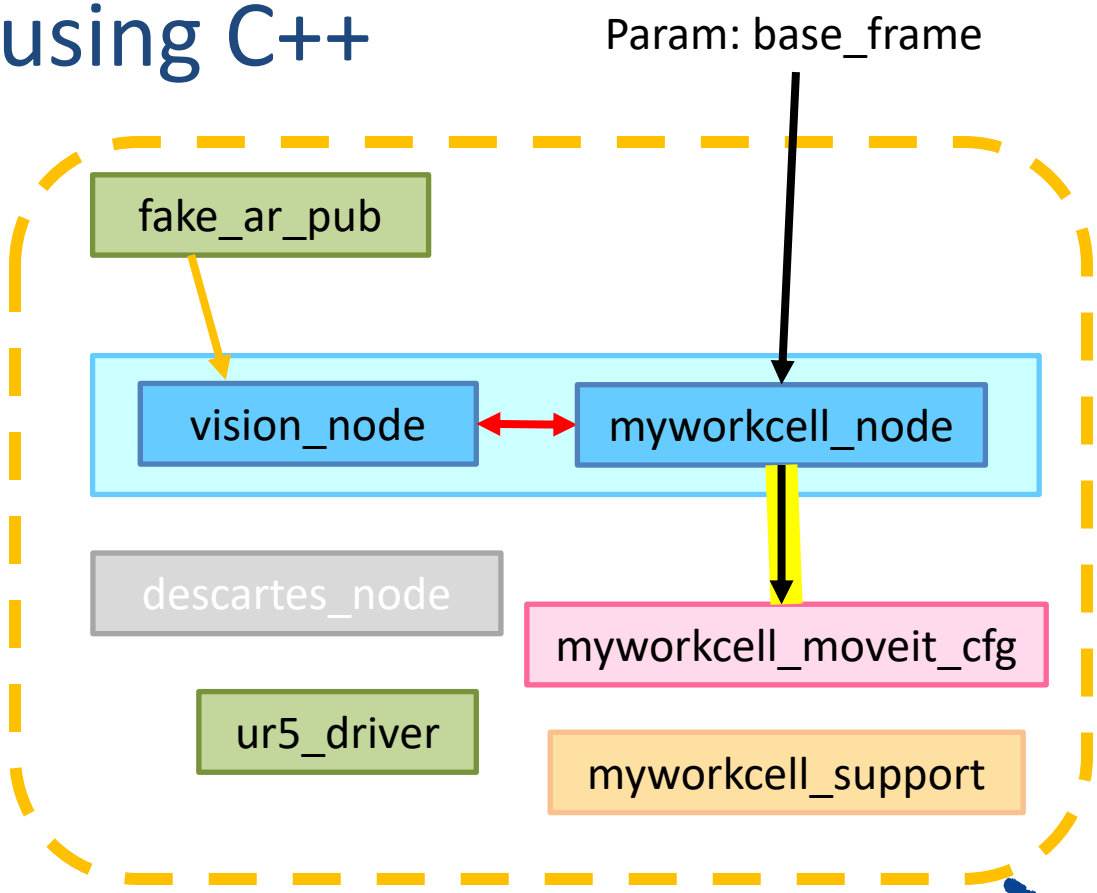
Cartesian position:

```
Affine3d pose = {x, y, z, r, p, y};
planner.setGoal(pose);
```

Exercise 4.0:

Motion Planning using C++





Common Motion Planners

Motion Planner	Application Space	Notes
OMPL / MoveIt	Free-space Planning	Stochastic sampling; Easy and convenient interface
TrajOpt	Trajectory Optimization	Optimize existing trajectory on constraints (distance from collision, joint limits, etc.)
Descartes	Cartesian path planning	Globally optimum; sampling-based search; Captures “tolerances”
CLIK	Cartesian path planning	Local optimization; Scales well with high DOF; Captures “tolerances”
STOMP	Free-space Planning	Optimization-based; Emphasizes smooth paths

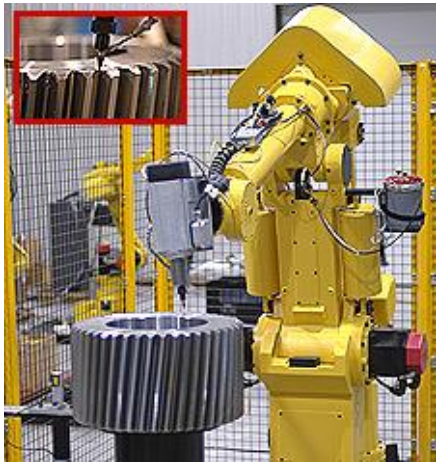
INTRODUCTION TO DESCARTES

Outline

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

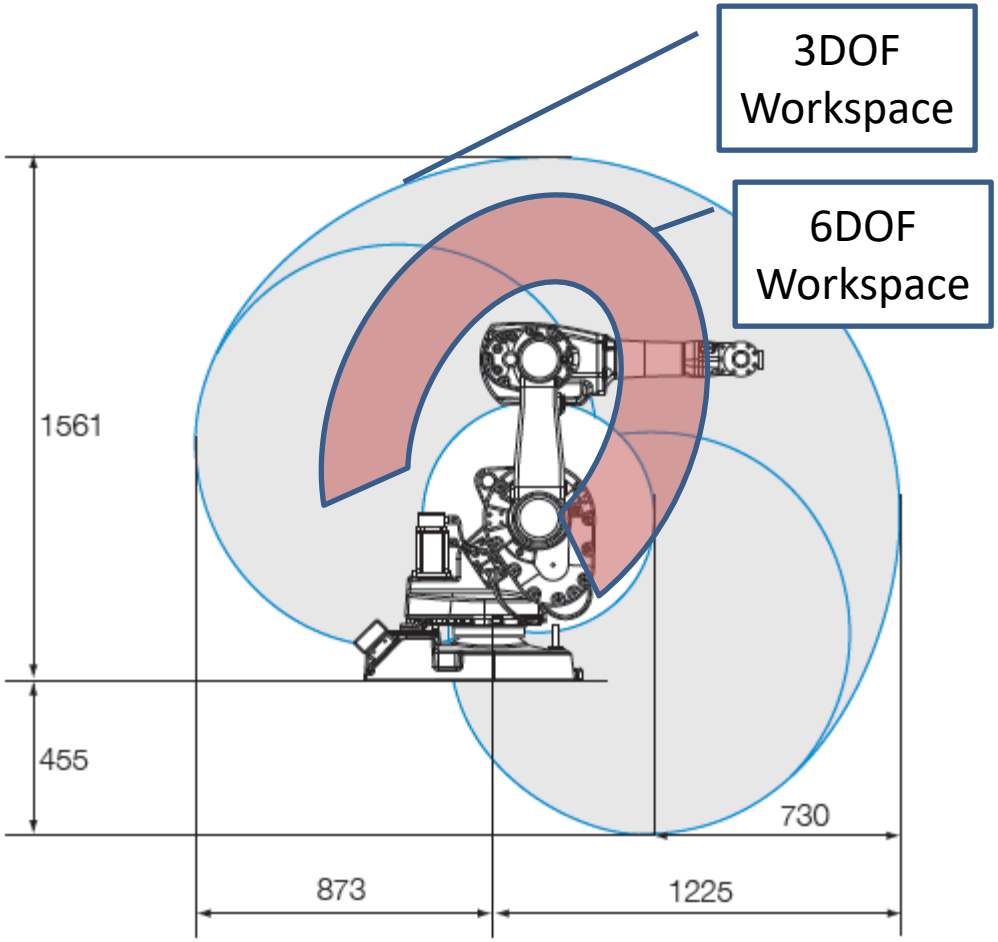
Introduction

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



Current Solution

- Arbitrary assignment of 6DOF poses, redundant axes -> IK
- Limited guarantee on trajectory timing
- Limitations
  - Reduced workspace
  - Relies on human intuition
  - Collisions, singularities, joint limits

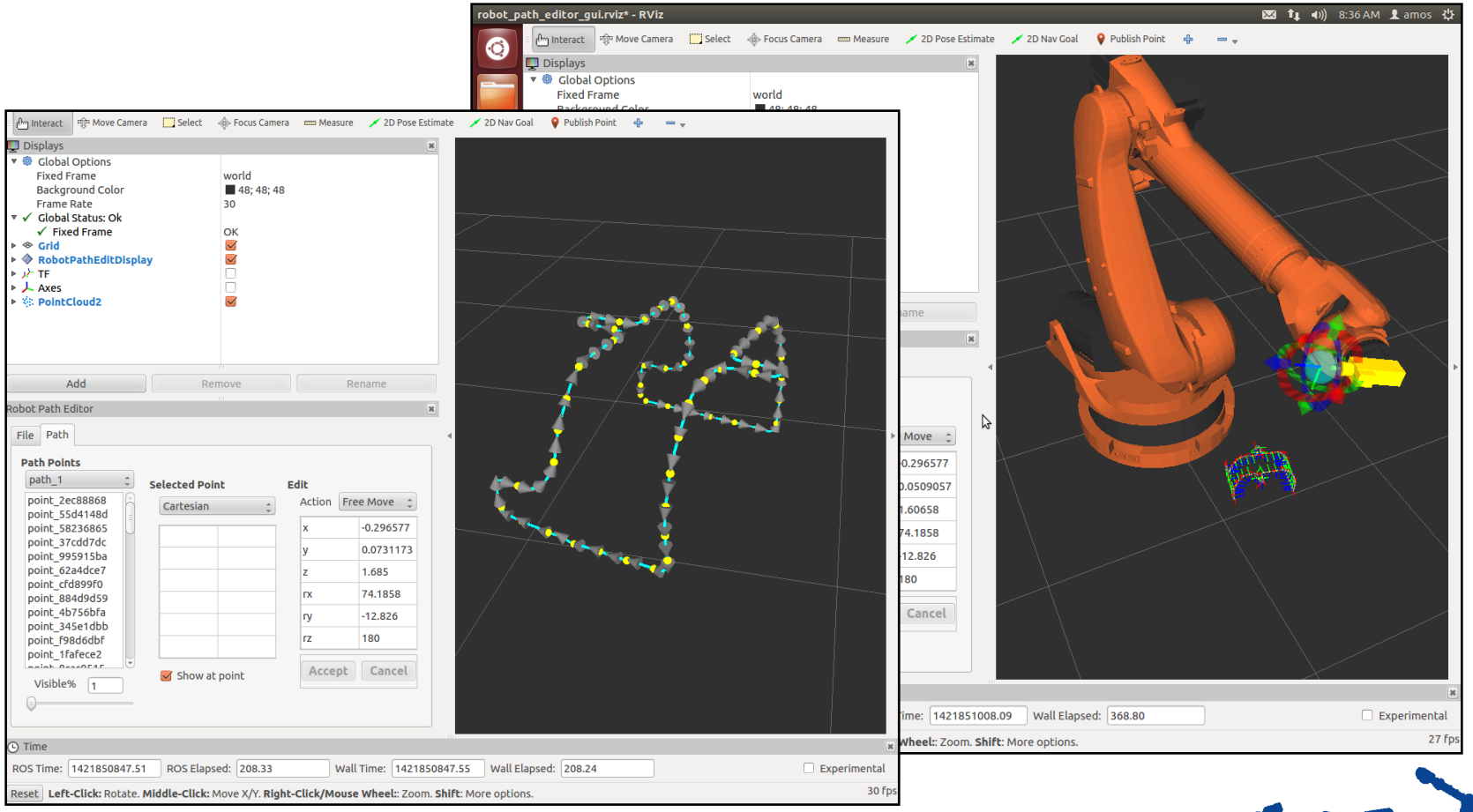


Descartes

- 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



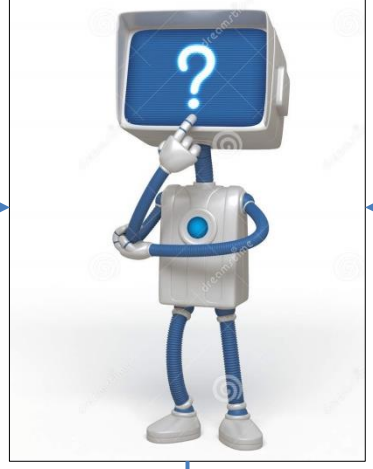
Open Source Details

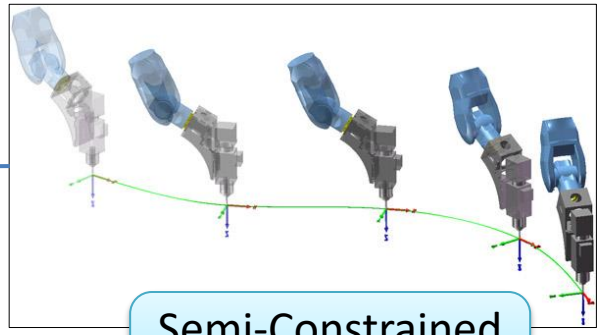
- Public development: https://github.com/ros-industrial-consortium/descartes
- Wiki Page: http://wiki.ros.org/descartes
- Acknowledgements:
  - Dev team: Dan Solomon (former SwRI), Shaun Edwards (former SwRI), Jorge Nicho (SwRI), Jonathan Meyer (former SwRI), Purser Sturgeon(SwRI)
  - Supported by: NIST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

Descartes Workflow

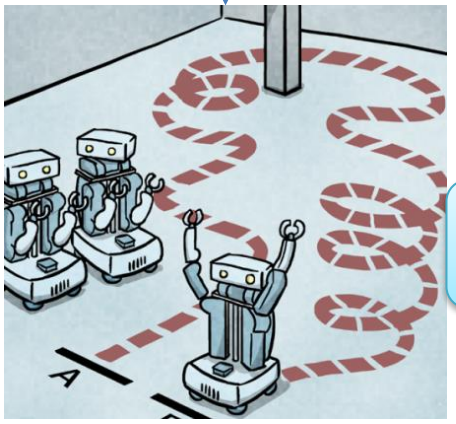
Robot Model

Path Planner





Semi-Constrained Trajectory



Robot Path

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

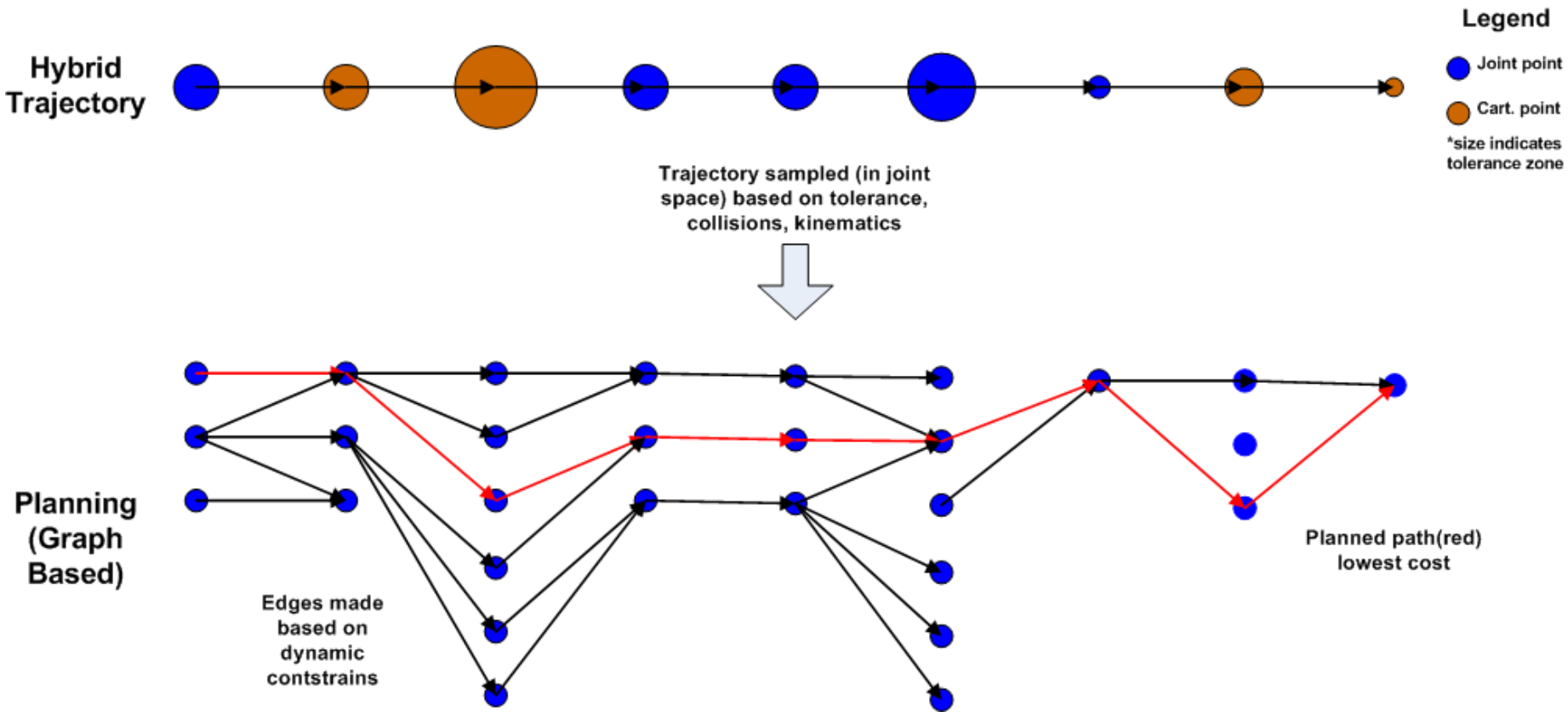
Descartes Interfaces

- 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

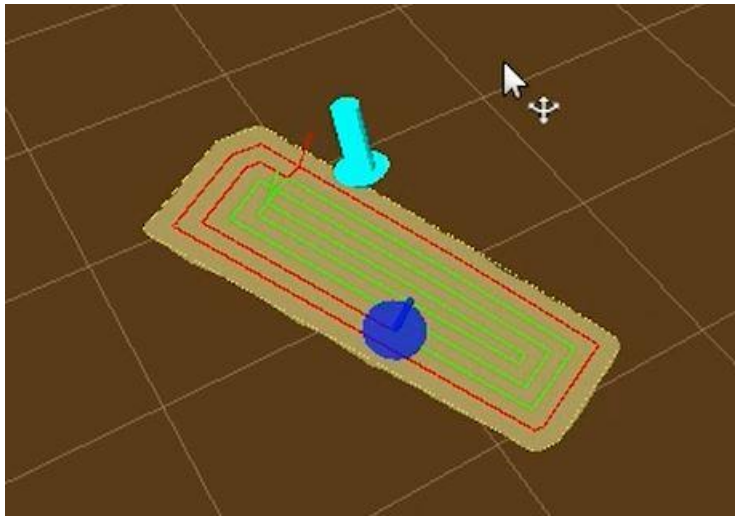
- Planners
  - Planners are the highest level component of the Descartes architecture.
  - Take a trajectory of points and return a valid path expressed in joint positions for each point in the tool path.
  - Two implementations
    - DensePlanner
    - SparsePlanner

Descartes Implementation

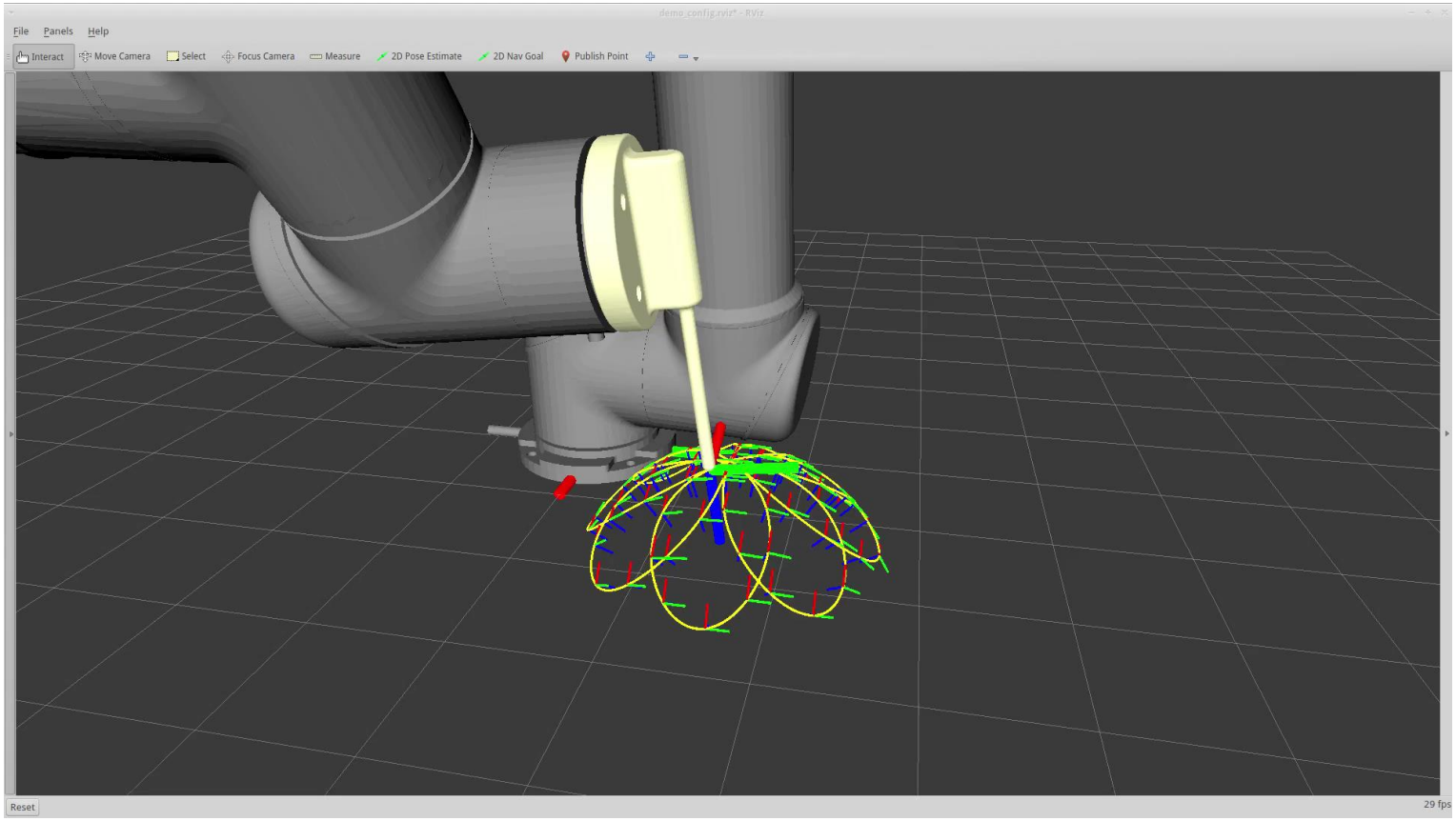


Descartes Input/Output

- 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)

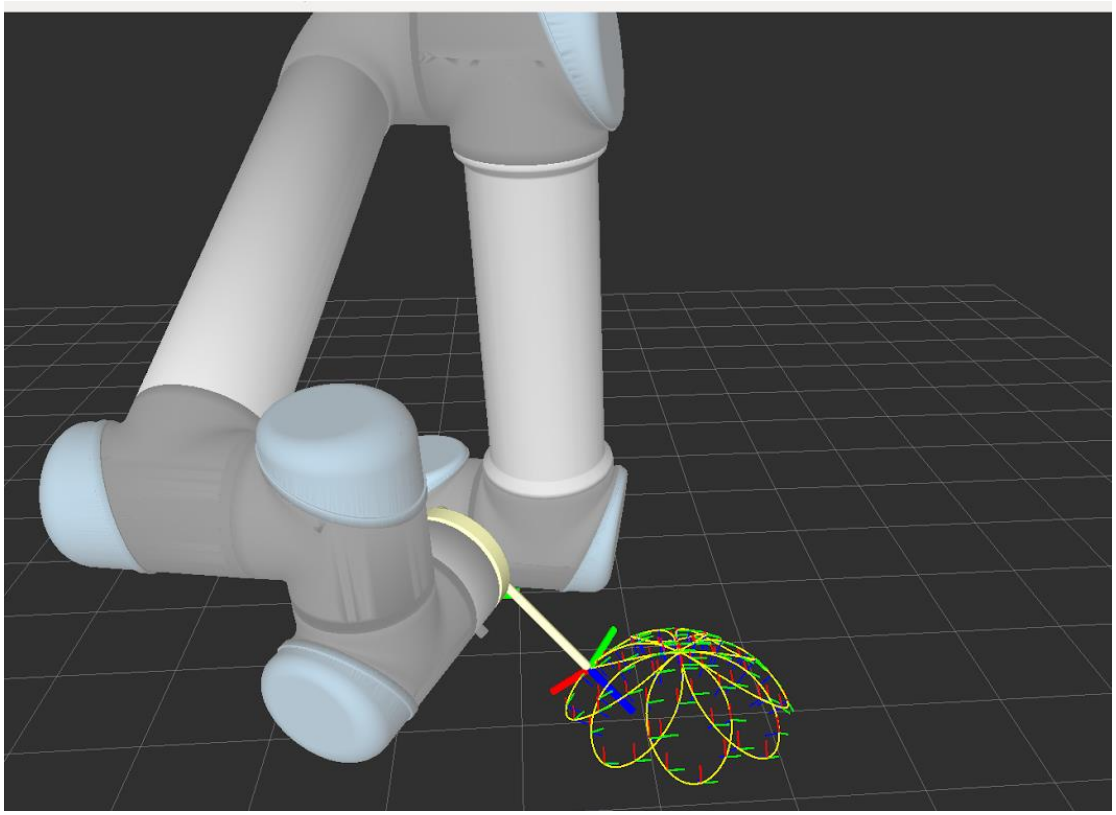


Descartes Demonstration



Exercise 4.1:

Descartes Path Planning



INTRODUCTION TO PERCEPTION

Outline

- Camera Calibration
- 3D Data Introduction
  - Exercise 4.2
- Explanation of the Perception Tools Available in ROS
- Intro to PCL tools
  - Exercise 4.2

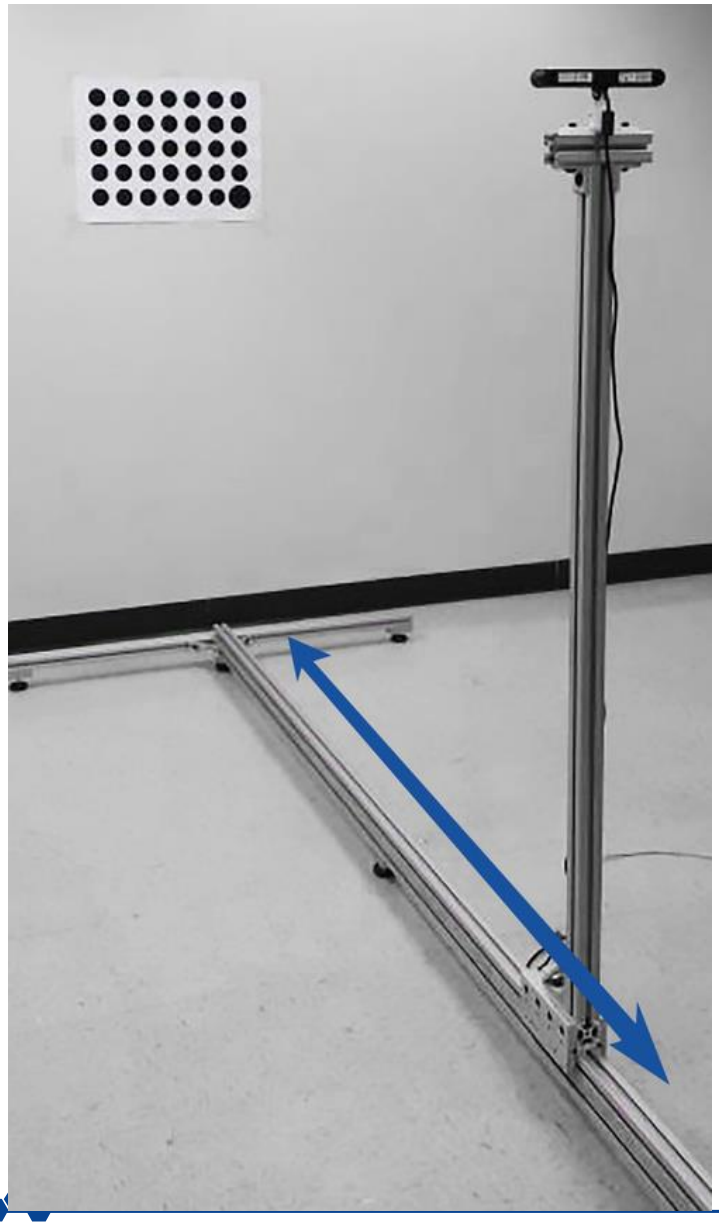
Objectives

- Understanding of the calibration capabilities
- Experience with 3D data and RVIZ
- Experience with Point Cloud Library tools*

Industrial Calibration

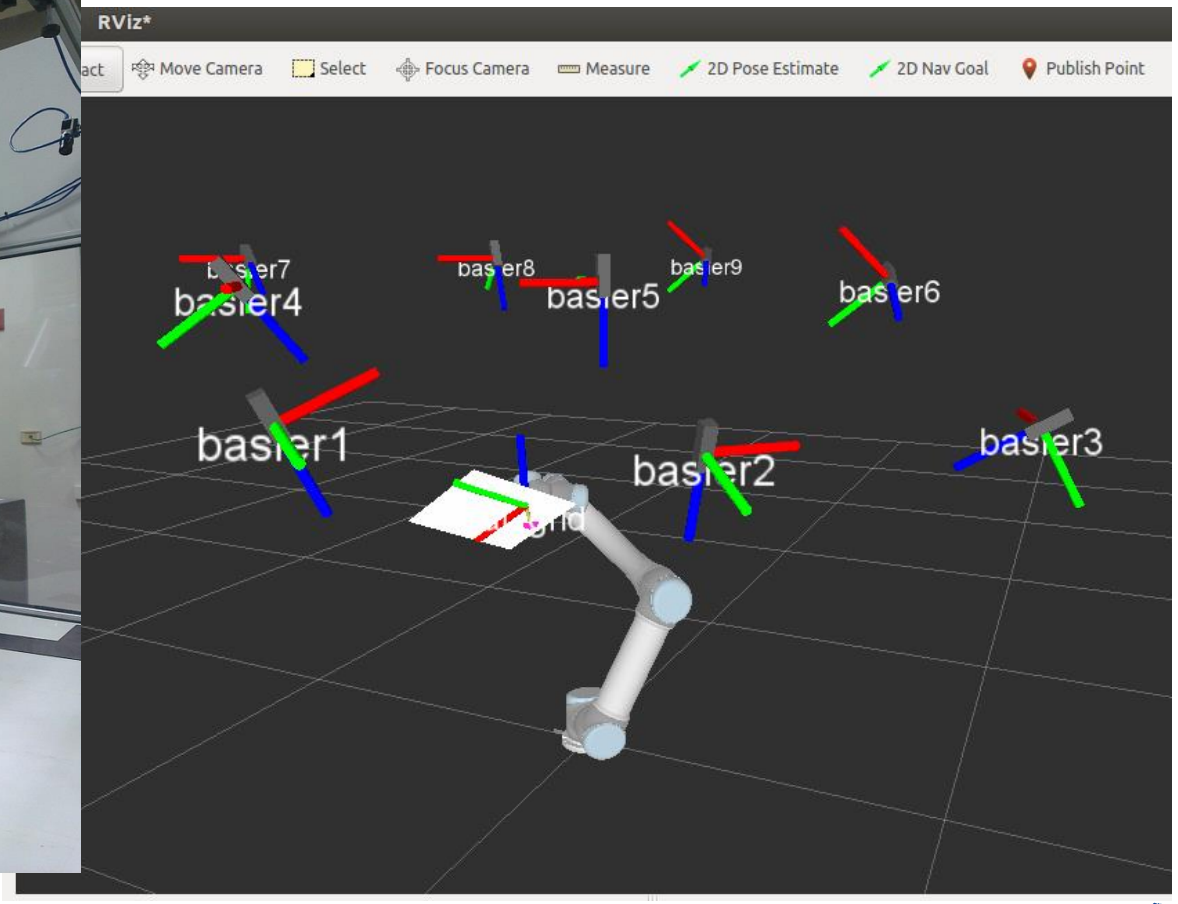
- Perform intrinsic and extrinsic calibration
- Continuously improving library
- Resources, library
  - Github link
  - Wiki link
- Resources, tutorials
  - Github industrial calibration tutorials link
  - Training Wiki link

Industrial (Intrinsic) Calibration



- The INTRINSIC Calibration procedure requires movement of the camera to known positions along an axis that is approximately normal to the calibration target.
- Using the resulting intrinsic calibration parameters for a given camera yields significantly better extrinsic calibration or pose estimation accuracy.

Industrial (Extrinsic) Calibration



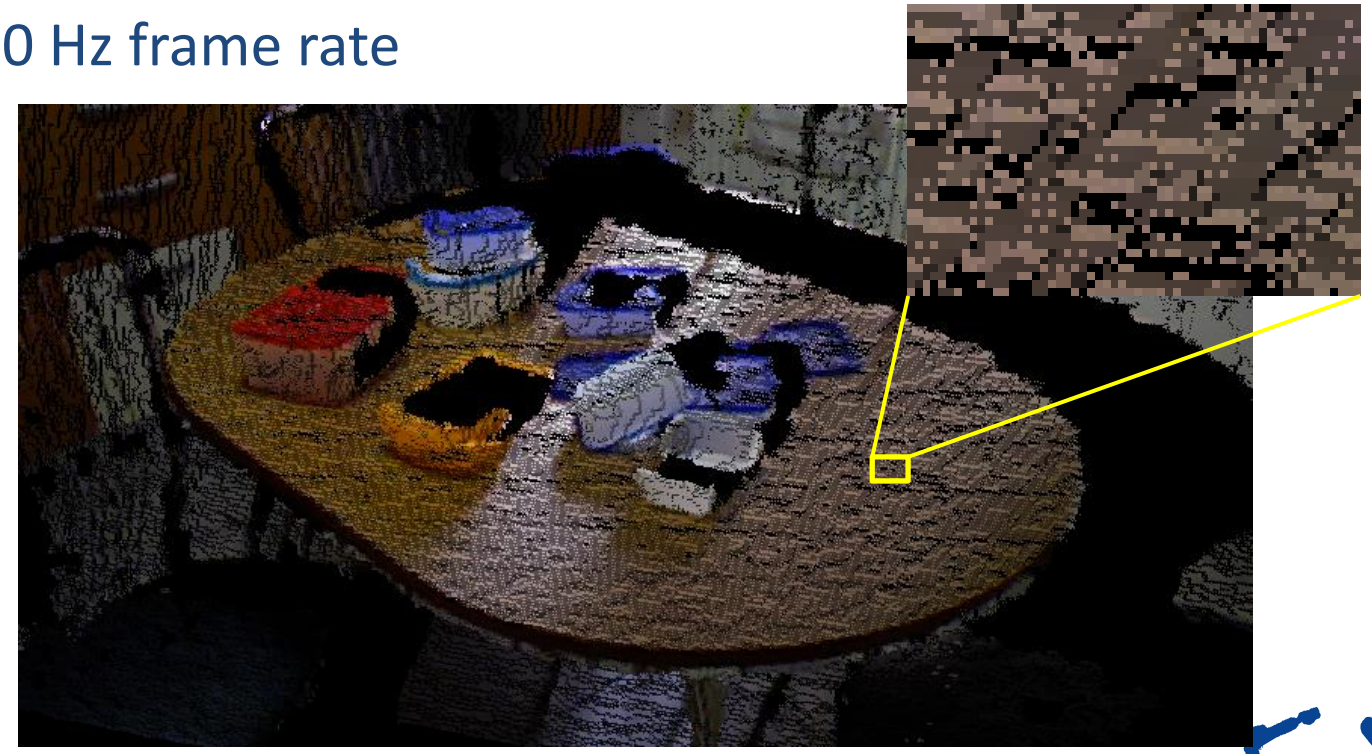
https://www.youtube.com/watch?v=MJFtEr_Y4ak

3D Cameras

- RGBD cameras, TOF cameras, stereo vision, 3D laser scanner
- Driver for Asus Xtion camera and the Kinect (1.0) is in the package openni_launch or openni2_launch
- Driver for Kinect 2.0 is in package iai_kinect2 (github link)
- https://rosindustrial.org/3d-camera-survey

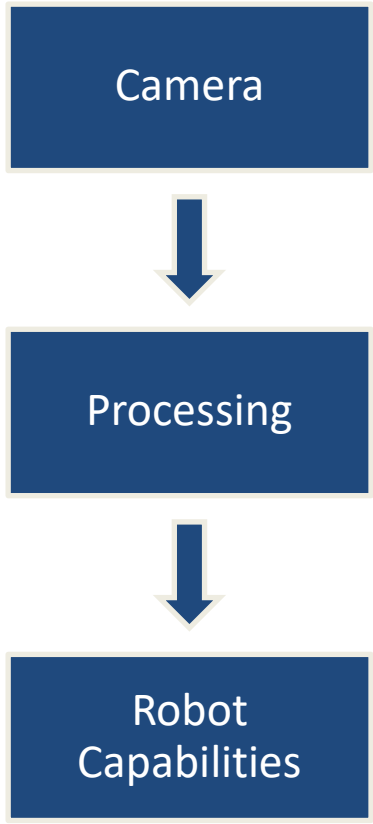
3D Cameras

- 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
- Process data in order to
  - Improve data quality ➡ filter noise
  - Enhance succeeding processing steps ➡ reduce amount of data
  - Create a consistent environment model ➡ Combine data from different view points
  - Simplify detection problem ➡ segment interesting regions
  - Gain knowledge about environment ➡ classify surfaces



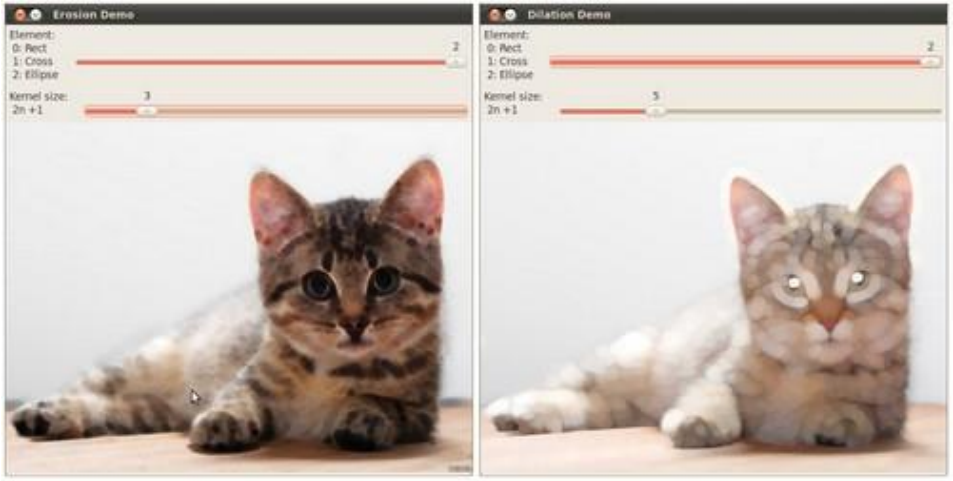
Perception Tools

- Overview of OpenCV
- Overview of PCL
- PCL and OpenCV in ROS
- Other libraries

- Focus on PCL tools for exercise

Perception Libraries (OpenCV)

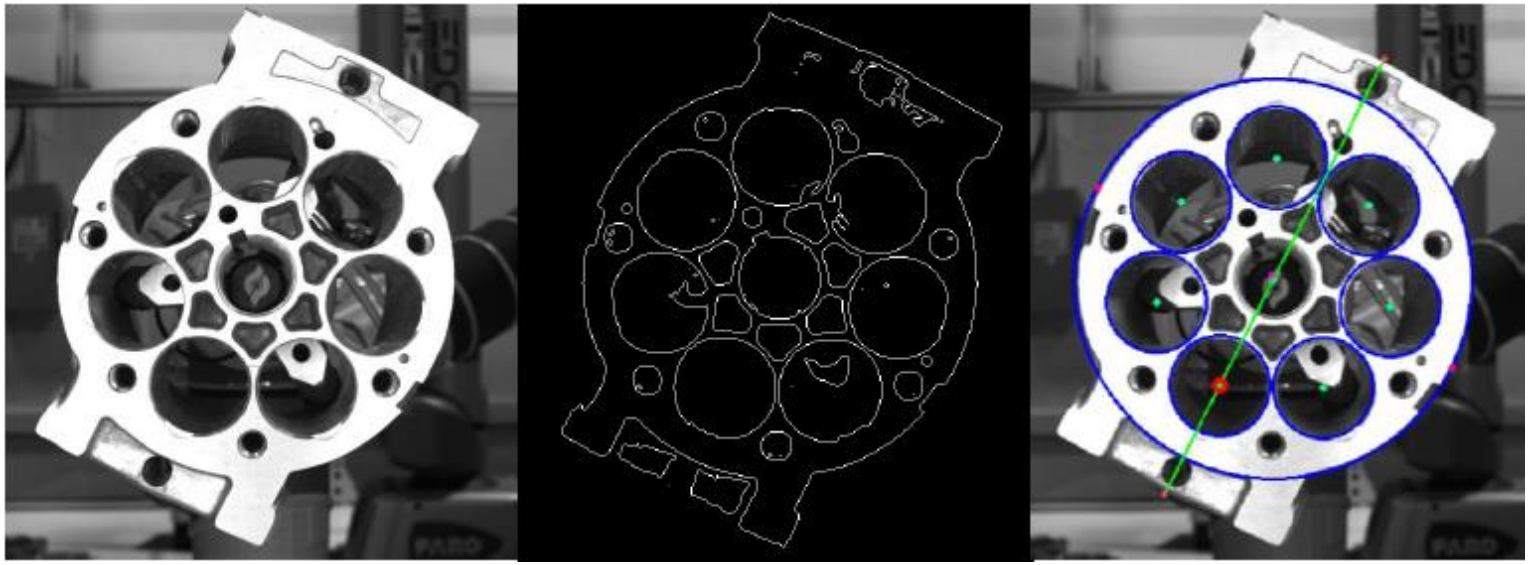
- Open Computer Vision Library (OpenCv) - http://opencv.org/
  - Focused on 2D images
  - 2D Image processing
  - Video
  - Sensor calibration
  - 2D features
  - GUI
  - GPU acceleration



http://opencv.org

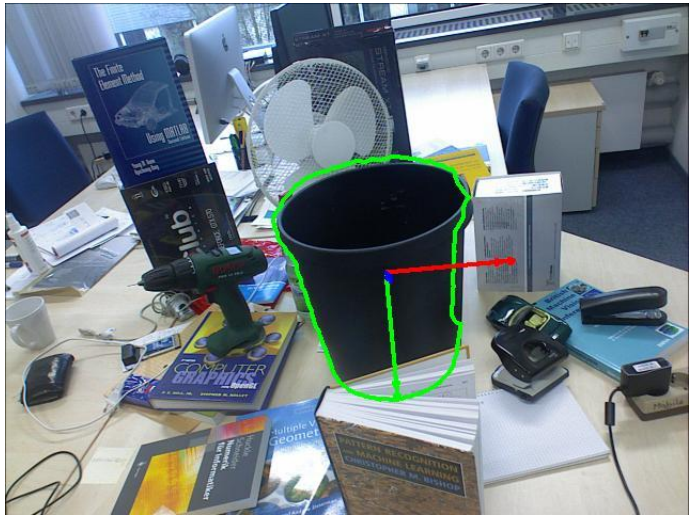
OpenCV tutorial

- Perform image processing to determine pump orientation (roll angle)
- Github tutorial link
- Training Wiki link



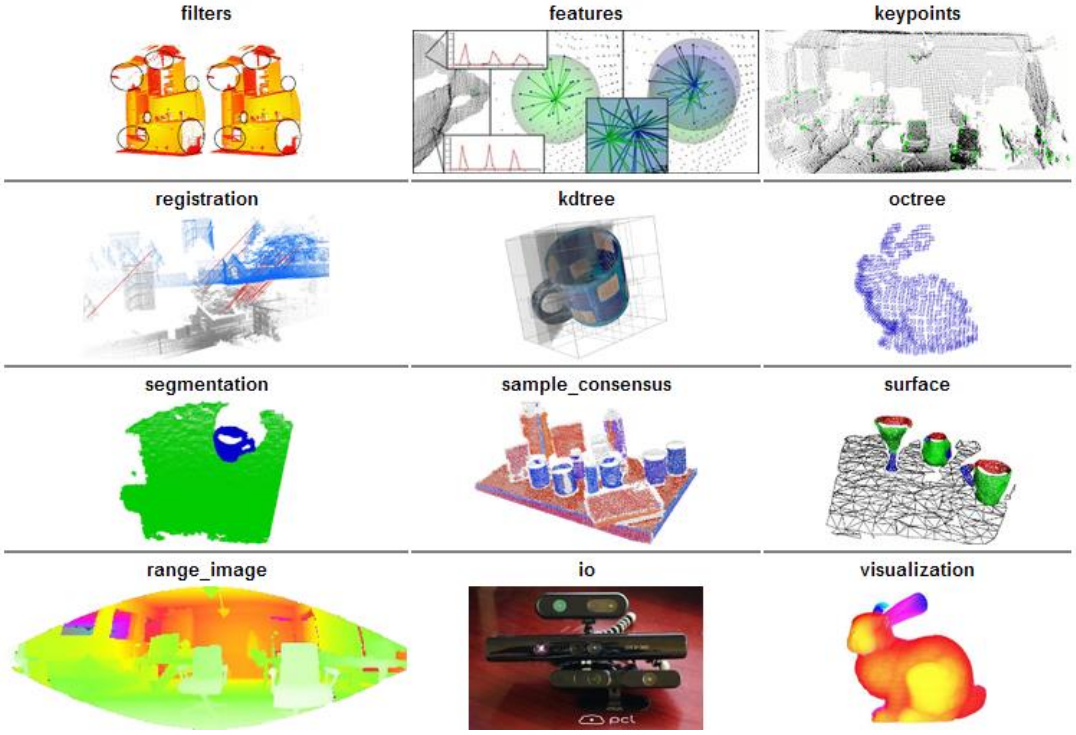
Perception Libraries (OpenCV)

- Open CV 3.2
  - Has more 3D tools
    - LineMod
      - https://www.youtube.com/watch?v=vsThfxzIUjs
    - PPF
  - Has opencv_contrib
    - Community contributed code
    - Some tutorials



Perception Libraries (PCL)

- Point Cloud Library (PCL) - http://pointclouds.org/
  - Focused on 3D Range(Colorized) data



http://pointclouds.org

Perception Libraries (PCL)

- PCL Command Line Tools
  - sudo apt install pcl-tools
  - Tools (140+)
    - pcl_viewer
    - pcl_point_cloud_editor
    - pcl_voxel_grid
    - pcl_sac_segmentation_plane
    - pcl_cluster_extraction
    - pcl_passthrough_filter
    - pcl_marching_cubes_reconstruction
    - pcl_normal_estimation
    - pcl_outlier_removal

ROS Bridges

- OpenCV & PCL are external libraries
- “Bridges” are created to adapt the libraries to the ROS architecture
  - OpenCV: http://ros.org/wiki/vision_opencv
  - PCL: http://ros.org/wiki/pcl_ros
    - Standard Nodes (PCL Filters): http://ros.org/wiki/pcl_ros#ROS_nodelets

Unfortunately, plc_ros has not yet been ported to ROS2.

Many More Libraries

- Many more libraries in the ROS Ecosystem
  - AR Tracker
    http://www.ros.org/wiki/ar_track_alvar
  - Robot Self Filter
    http://www.ros.org/wiki/robot_self_filter

Exercise 4.2

- Play with PointCloud data
  - Play a point cloud file to simulate data coming from a Asus 3D sensor.
  - Matches scene for demo_manipulation
  - ~~3D Data in ROS~~ (plc_ros not yet ported to ROS2)
  - Use PCL Command Line Tools
- https://github.com/ros-industrial/industrial_training/wiki/Introduction-to-Perception

Review/Q&A

Session 3

ROS-Industrial

- Architecture
- Capabilities

Motion Planning

- Examine MoveIt Planning Environment
- Setup New Robot
- Motion Planning (Rviz)
- Motion Planning (C++)

Session 4

Descartes

- Path Planning
- Trajectory points

Perception

- Calibration
- PointCloud File
- OpenCV
- PCL
- PCL Command Line Tools