The Industrial Trajectory Generation and Python API of pilz_industrial_motion

https://wiki.ros.org/pilz_robots

3rd ROS Meet-Up
Stuttgart, February 6 2020

Christian Henkel
Advanced Development
Two talks

pilz_robots
Drivers
Hardware Support
→ ROScon

pilz_industrial_motion
Planner
API
→ NOW
Recap of „Safety Certified ROS-native Industrial Manipulator“ @ ROSCon

Traditional Setup

- ROS would be merely an afterthought

Intended Setup

- ROS as core component
Example Application: Visual Inspection

Task: Inspect part features for large number of product variants
Approach: Robot on-board camera supported on database to lookup poses and save results
Strengths of ROS:
- High-level control based on the adaption of State-Machine packages
- Interface with other software components
- Use of workspace based (OMPL) and deterministic (pilz_industrial_motion) motion planners
FTP Industrial Trajectory Generation for MoveIt!

Goal:
- Reproducible trajectories (PTP, LIN, CIRC)
- Fast computation
- Easy-to-use interface
  - Motion from RViz
  - Programming with Python API
  - Tutorials

Working for every robot which has a moveit_config.

Supported by ROSin - ROS-Industrial Quality-Assured Robot Software Components. More information: rosin-project.eu

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 732287.
**Strengths of MoveIt!**

The framework combines:

- Kinematics module(s)
- Collision checking with the environment model
- Trajectory execution

**Diagram:**

- **User Interface**
  - RViz or Python or...

- **Goal:**
  - MoveGroupAction(MotionPlanRequest)

- **Robot Controller(s)**
- **move_group**
- **Inverse Kinematics**
- **Joint Limits**
- **Environment Model**
- **Pilz CommandPlanner**
- **other capabilities...**
- **JointTrajectoryAction**
Motion Types

LIN
Linear intrapolation in cartesian space

PTP
Linear intrapolation in joint space

CIRC
Circular intrapolation in cartesian space
Pilz CommandPlanner

- Trapezoidal velocity profiles
- Collision checking (no avoidance)
- Blend combines a sequence of commands: e.g. LIN-LIN
User-Interface: Python-API

- Easy-to-use
- Versioning
- Move Command
- Reference Poses or Joint Values
- Relative Motions
- Sequences with Blending

```python
r = Robot(__REQUIRED_API_VERSION__)

# Simple ptp movement
r.move(Ptp(goal=[0, 0.5, 0.5, 0, 0, 0], vel_scale=0.4))

start_joint_values = r.get_current_joint_states()

# Relative ptp movement
r.move(Ptp(goal=[0.1, 0, 0, 0, 0, 0], relative=True, vel_scale=0.2))
r.move(Ptp(goal=Pose(position=Point(0, 0, -0.1)), relative=True))
r.move(Ptp(goal=[-0.2, 0, 0, 0, 0, 0], relative=True, acc_scale=0.2))

sequence = Sequence()
sequence.append(Lin(goal=Pose(position=Point(0.2, 0, 0.8)), vel_scale=0.1, acc_scale=0.1))
sequence.append(Circ(goal=Pose(position=Point(0.2, -0.2, 0.8)), center=Point(0.1, -0.1, 0.8), acc_scale=0.4))
sequence.append(Ptp(goal=pose_after_relative, vel_scale=0.2))
```

https://github.com/PilzDE/pilz_industrial_motion/blob/melodic-devel/pilz_robot_programming/examples/demo_program.py
With a focus on quality

Documentation
- Overview on wiki.ros.org/pilz_robots
- Tutorials
- API-Documentation

Tests
- Unit- and Integration tests (Travis-CI-Integration)
- ~100% code coverage
Summary / Outlook

Industrial Trajectory Generation
- LIN, PTP, CIRC
- Blending

Python API
- Easy to use
- Versatile

Example using two planners
- ompl + Pilz
- LIN to approach
- ompl in free space

MoveIt 2.0!

World MoveIt Day