cartesian_controllers

Stefan Scherzinger

FZI Research Center for Information Technology

ROSCon 2019, Macao

November, 1st
Why this package?

- Closed loop force control
- Direct teaching
- Contact-rich manipulation
- Manual guidance
- Tele-operation

- You want task space control
- You don’t need collision checking or planning
- You want to use ROS-control

---
cartesian_controllers
You have:

- Joint position/velocity streaming interface
- ROS control HW abstraction
- This cartesian_controllers package
- Application with real-time end effector control

Robot

ROS control HW abstraction
- VelocityJointInterface
- PositionJointInterface
- EffortJointInterface

cartesian_controllers
- cartesian_motion_controller
- cartesian_force_controller
- cartesian_compliance_controller
- cartesian_controller_handle

Other controllers
- Joint Trajectory Controller

User application
Three main controllers

- **cartesian_motion_controller**
  - You want to follow a moving target
  - The targets might be sparsely sampled
  - You prefer smoothness over accuracy

- **cartesian_force_controller**
  - You want to control the robot with a wrench in contacts
  - You have a wrist ft sensor

- **cartesian_compliance_controller**
  - You want to follow a moving target
  - You want to react to external disturbances
  - You have a wrist ft sensor
How do they work?

- Apply PD gains and interpret this as a wrench
- Mapping from wrench to joint space
- Iterative, forward dynamics solver, based on virtually conditioned twin

\[ \epsilon \xrightarrow{\frac{d}{dt}} \dot{\epsilon} \xrightarrow{K_p} + \xrightarrow{K_d} f \]

\[ f \rightarrow H^{-1}J^T \rightarrow \dot{q} \rightarrow \int \rightarrow \int \rightarrow \text{Robot} \]
How to use them?

1. Before startup
   - Kinematic chain (base, tip)
   - Sensor & compliance frame
   - Controllable joints

2. Startup and switching
   - Controller manager

3. Online Configuration
   - Gains, stiffness, solver

4. Control interfaces
   - **cartesian_motion_controller**
     - User target: geometry_msgs/PoseStamped
   - **cartesian_force_controller**
     - User target: geometry_msgs/WrenchStamped
   - **cartesian_compliance_controller**
     - User target: geometry_msgs/PoseStamped
     - Sensor input: geometry_msgs/WrenchStamped
Recent works using cartesian_controllers

- Teleoperation with user motion tracking
- Pilz PRBT

**cartesian_motion_controller**

- Deep-learned contact skill model with force control
- Universal Robots UR10

**cartesian_force_controller**

- Add-on compliance control for door sealing assembly
- KUKA KR16

**cartesian_compliance_controller**

Further reading

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>IK solving for sparse targets</td>
<td>arXiv: 1908.06252</td>
</tr>
<tr>
<td>Contact skills with force control</td>
<td>arXiv: 1908.06272 (IROS 2019)</td>
</tr>
<tr>
<td>Initial idea</td>
<td>DOI: 10.1109/IROS.2017.8206325</td>
</tr>
</tbody>
</table>
Summary

You have

- Joint position/velocity streaming interface
- ROS control pipeline
- Application with real-time, direct, task space control

You can use:

- cartesian_controllers
  - cartesian_motion_controller
  - cartesian_force_controller
  - cartesian_compliance_controller

You can find more information and resources at:

github.com/fzi-forschungszentrum-informatik/cartesian_controllers

Thank you

github.com/fzi-forschungszentrum-informatik/cartesian_controllers

scherzin@fzi.de