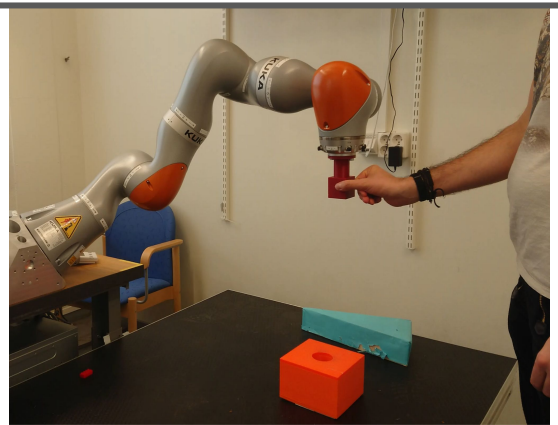
Kraftregelung von Roboterarmen mit ROS 2

Matthias Mayr, Freelancer



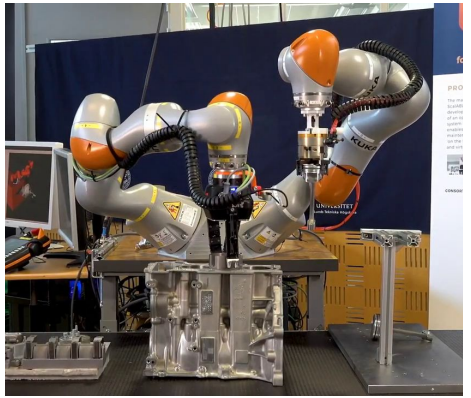
Über mich:

Robotiker im Bereich von flexiblen Industrieaufgaben

… und die Motivation

- Aufgabenplanung und Wissensrepräsentation
- Robotergenerische Lösungen
- Kontaktreiche Aufgaben
- Verstärkendes Lernen in Simulation

→Eigene Kraftregelung war notwendig



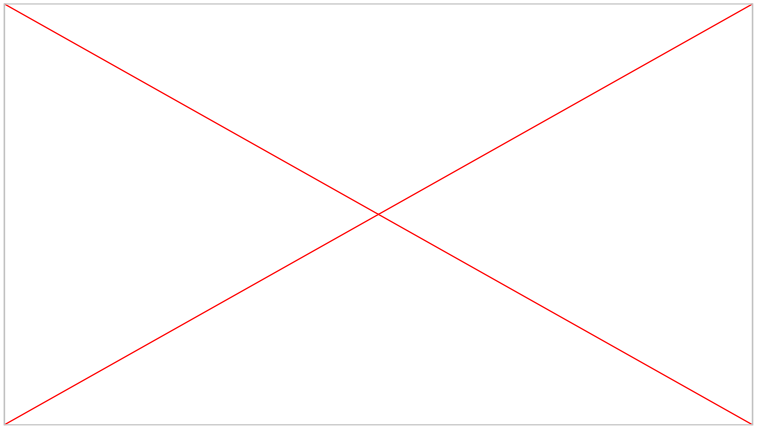
Einstieg in die Regelung war nicht einfach.

Warum Kraftregelung?

- Aufgaben mit Umgebungskontakt
  - Oberflächenbearbeitung
  - Steck- und Schraubaufgaben

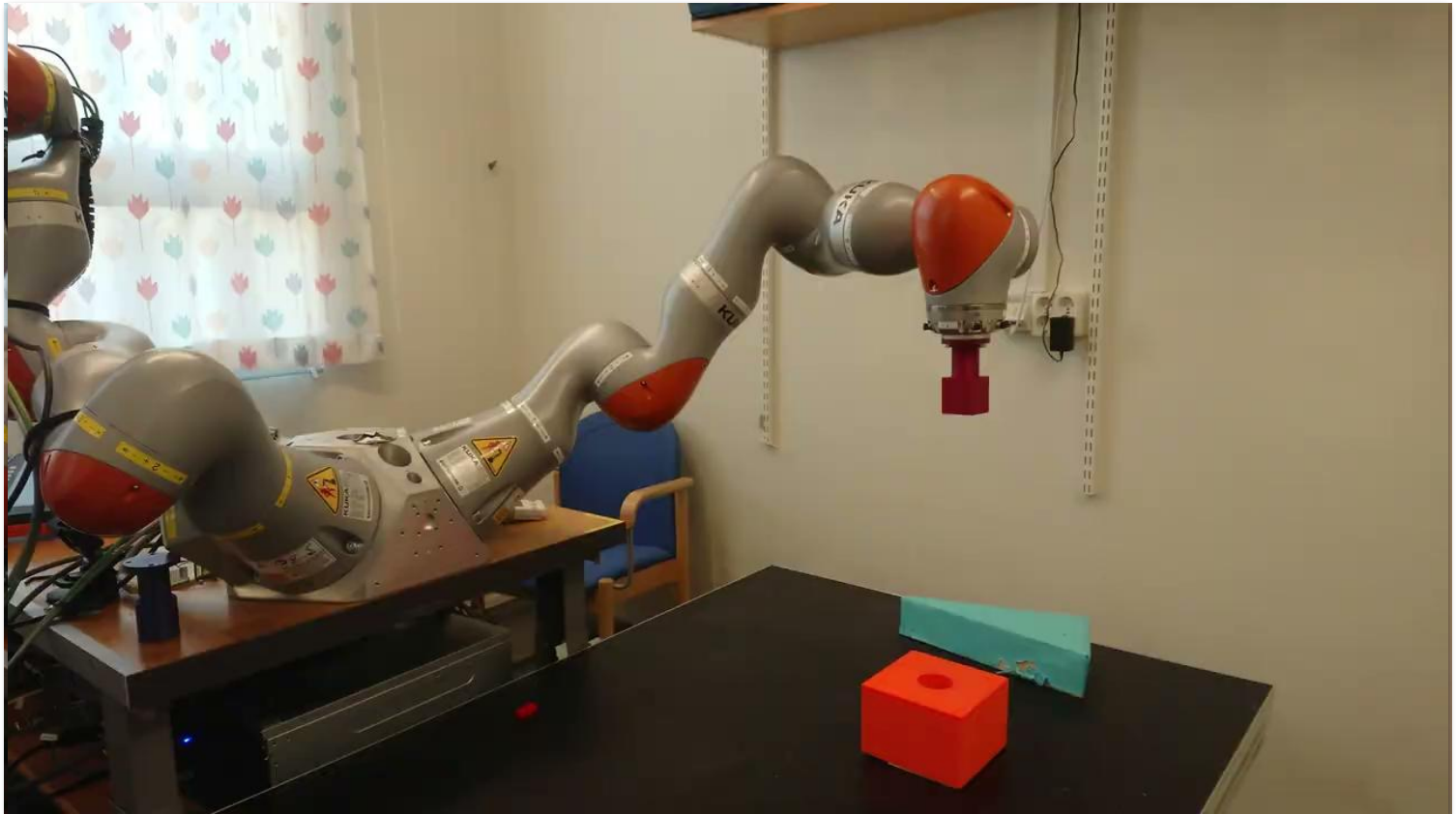
- Anlernen oder Korrektur von Aufgaben
  - *“Programmieren durch Vormachen”*
  - *Teleoperation*

- Unsichere Umgebungen
  - Objektpositionen variieren

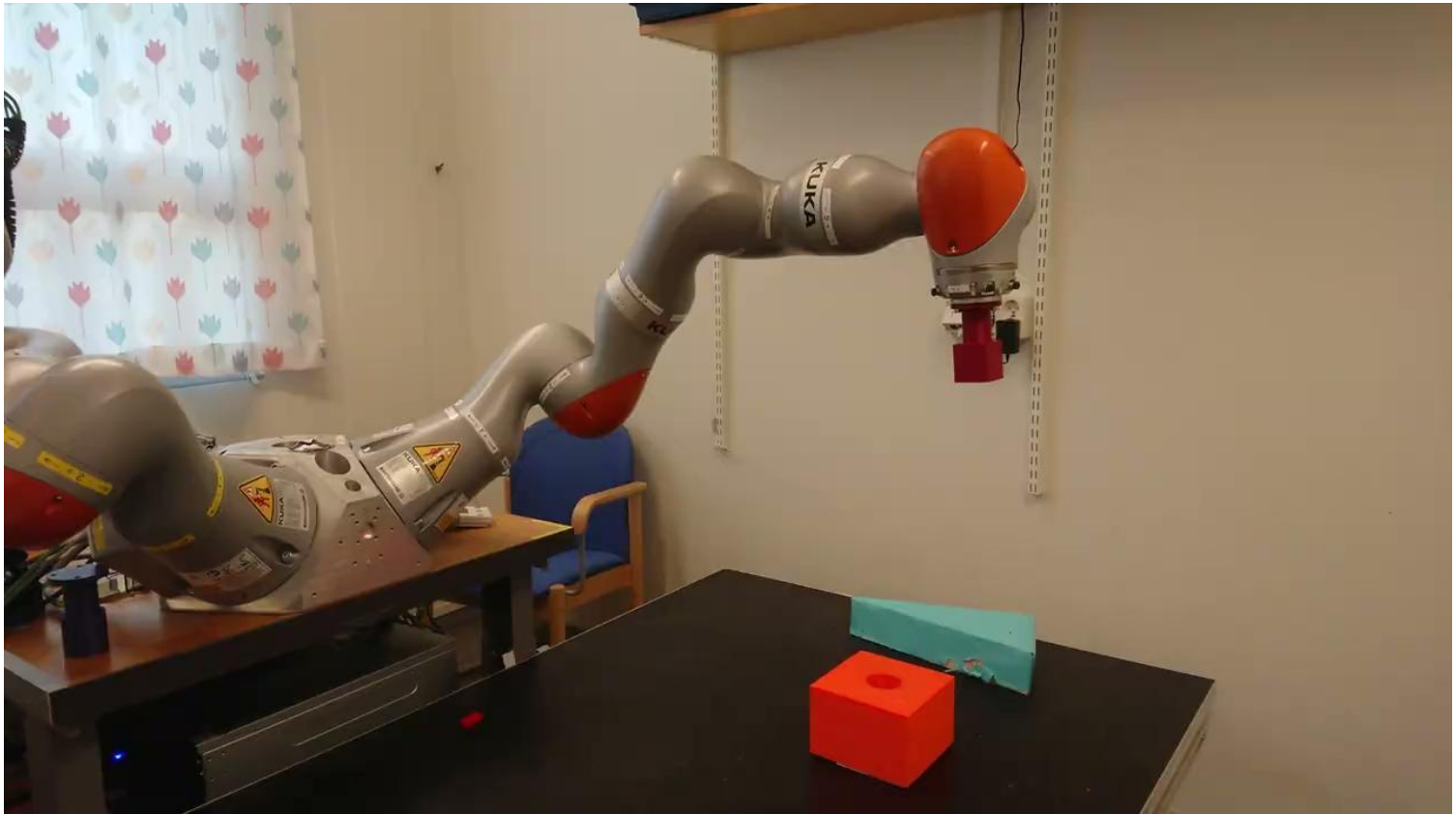


Fähigkeiten von Kraftregelung

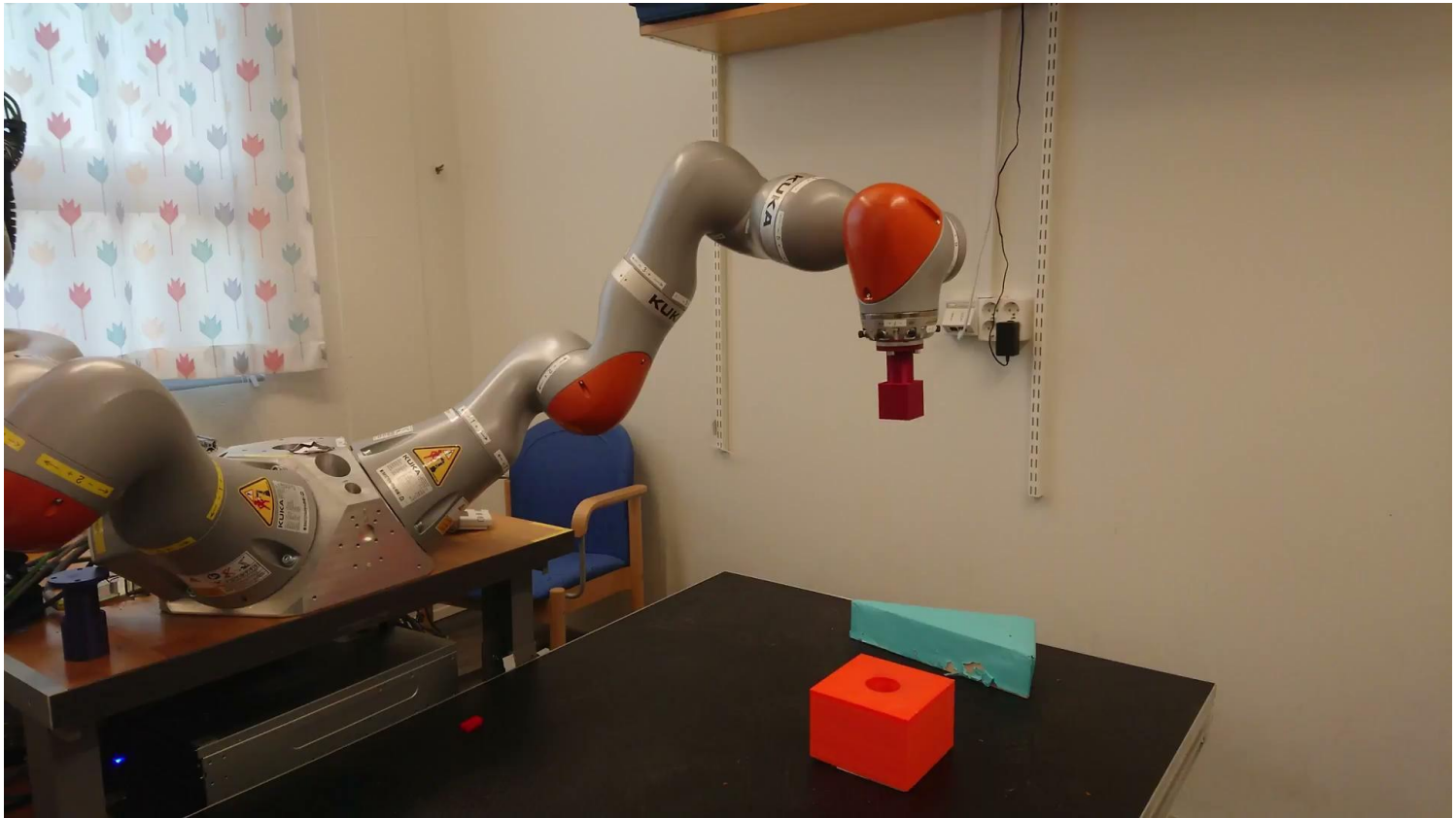
Kartesische Regelung



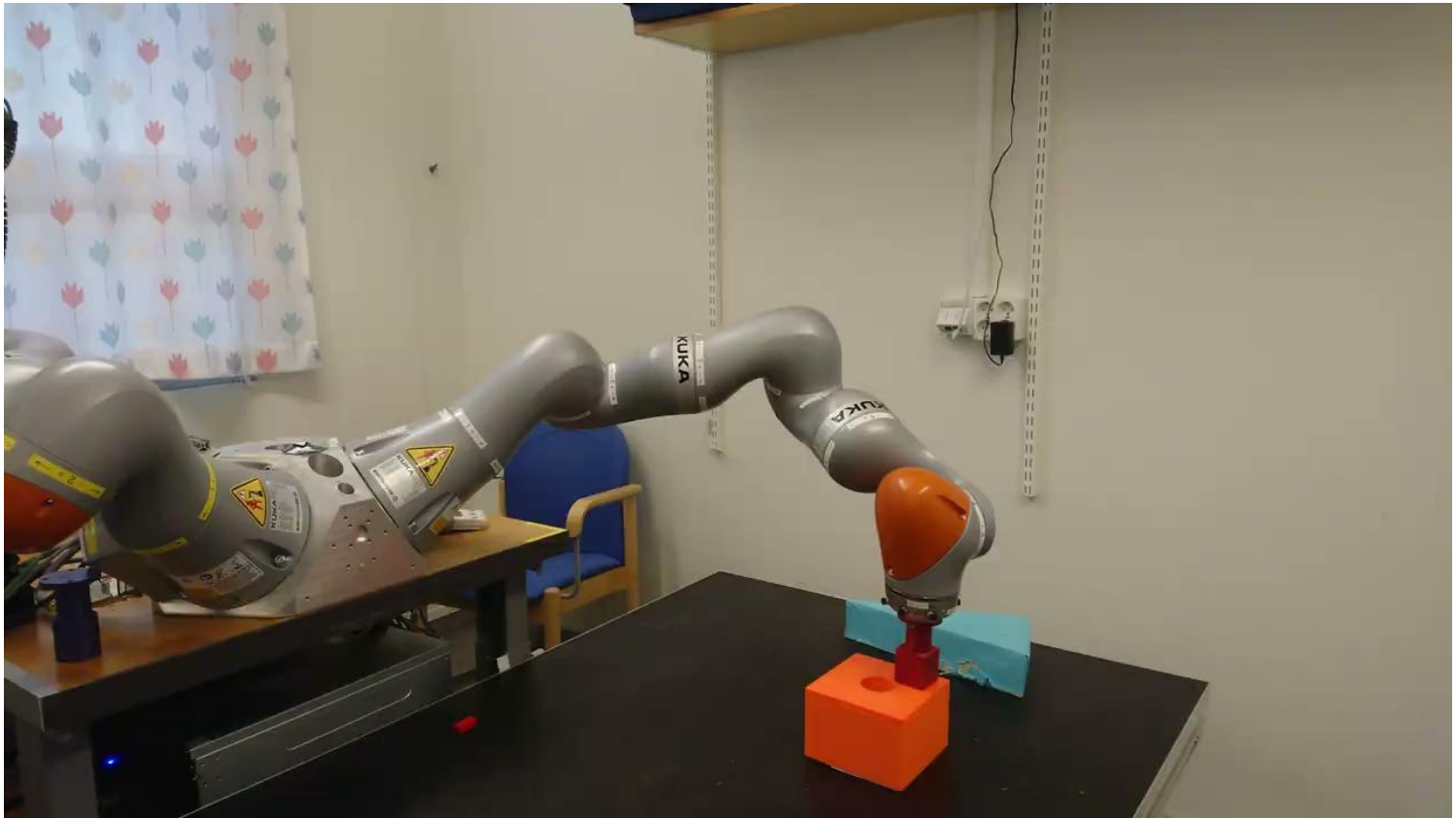
Keine translatorische Steifigkeit



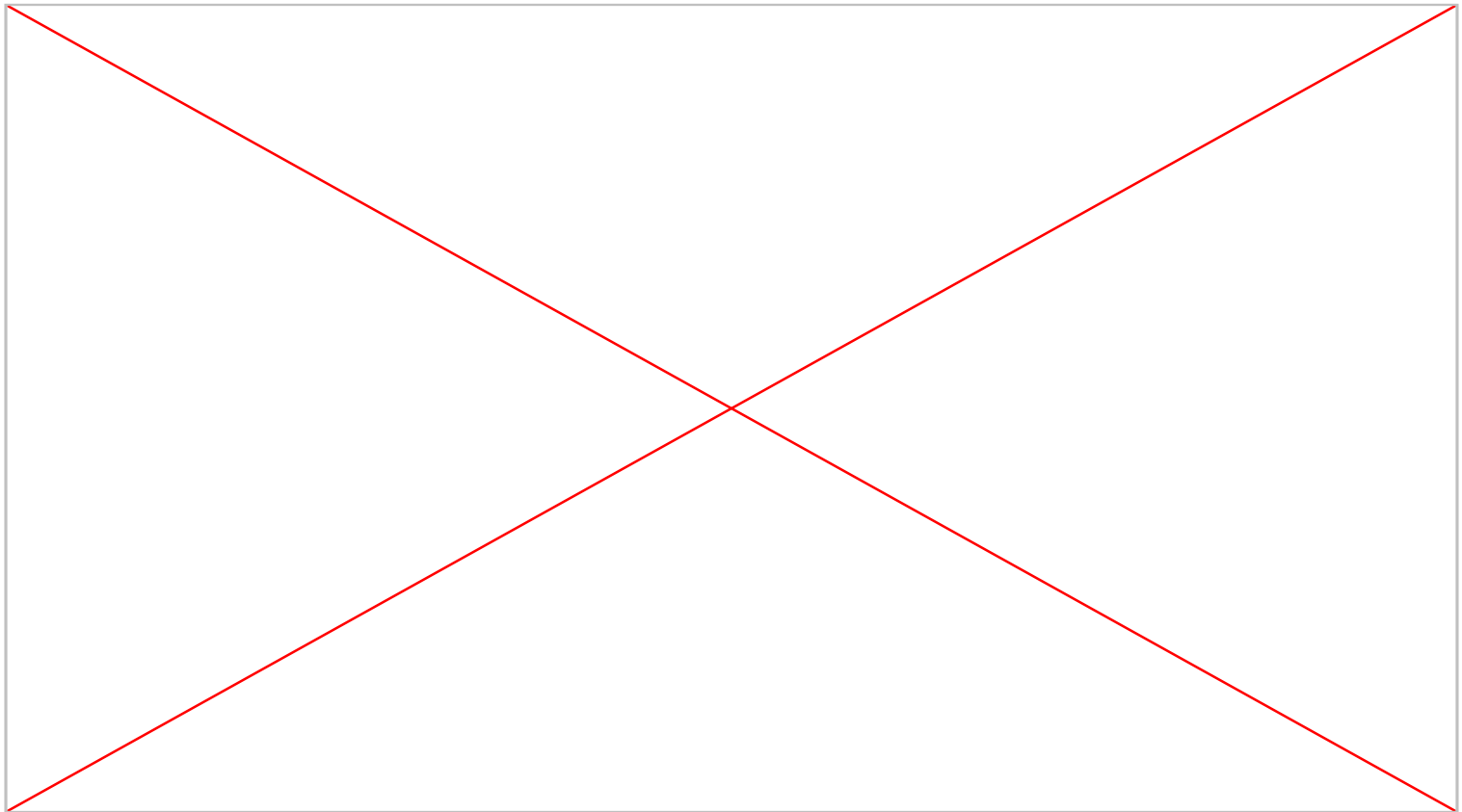
Keine rotatorische Steifigkeit



Krafteinbringung mit dem Werkzeug



Trajektorienausführung



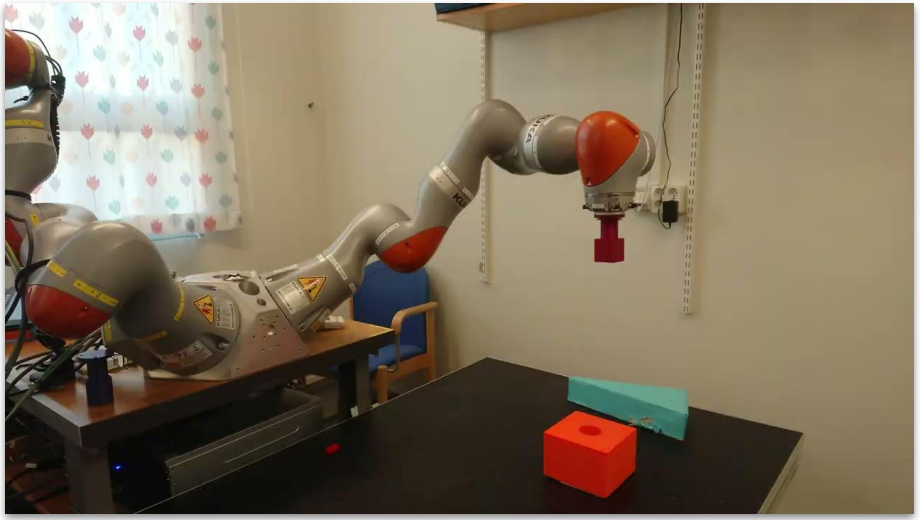
Kraftregelungsarten und Roboterschnittstellen

Arten von Kraftregelung

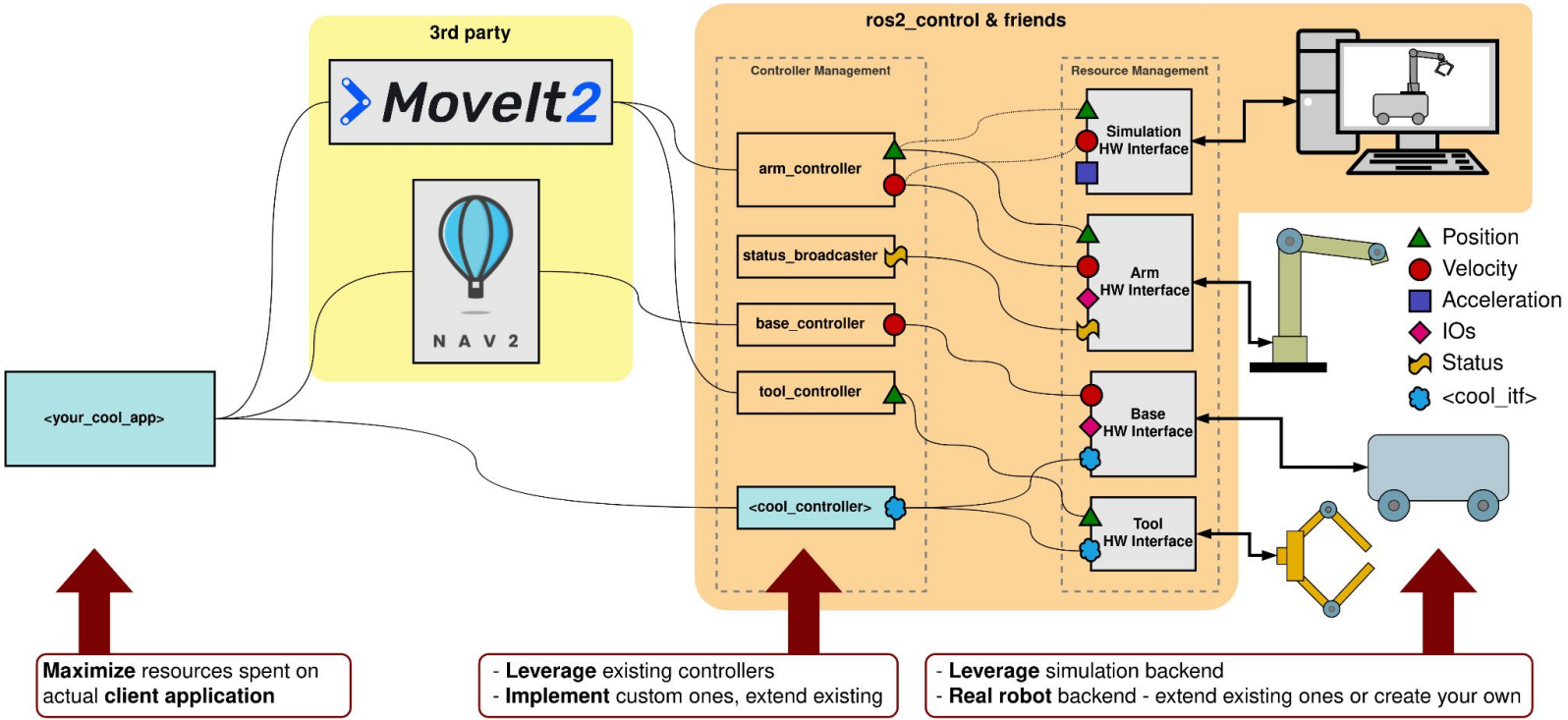
- **Kartesische Regelung - Werkzeug**
  “Cartesian Impedance Control”
  “Admittance Control”

- **Gelenkregelung**
  “Joint Impedance Control”

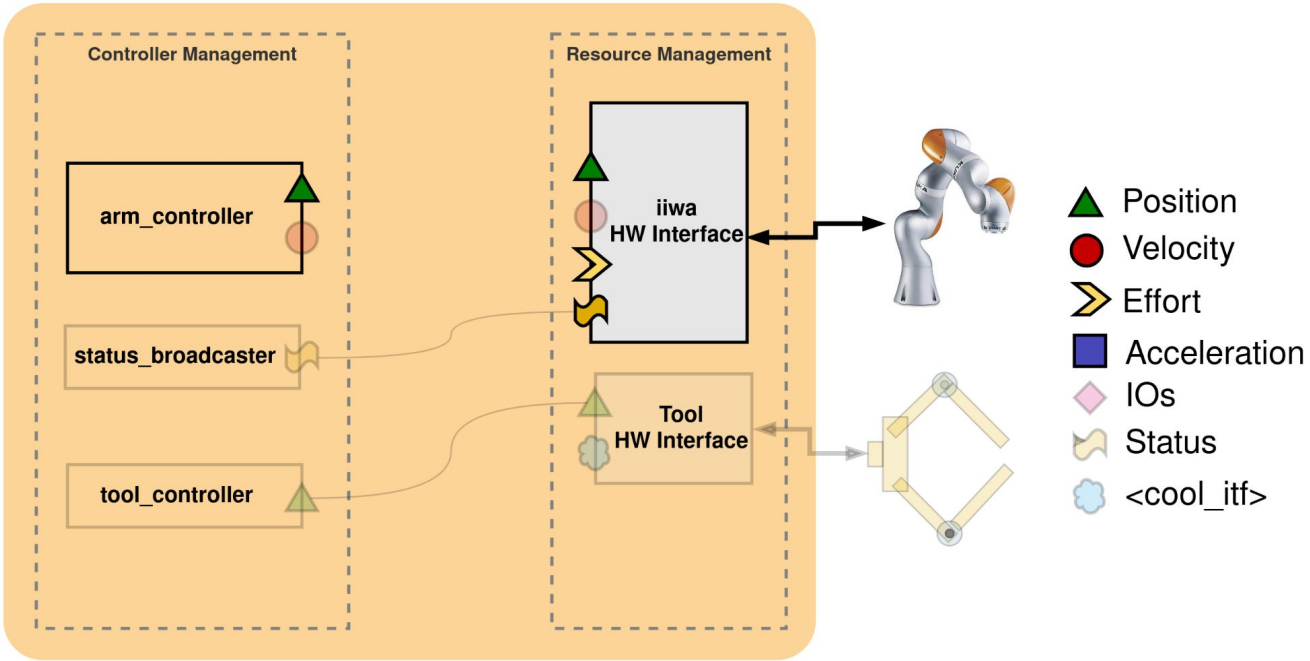
- **Spezialformen**
  URs “Force Mode”



Kommandoschnittstellen von Robotern



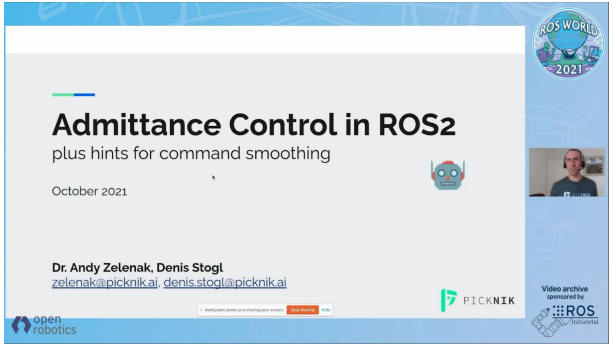
Kommandoschnittstellen von Robotern

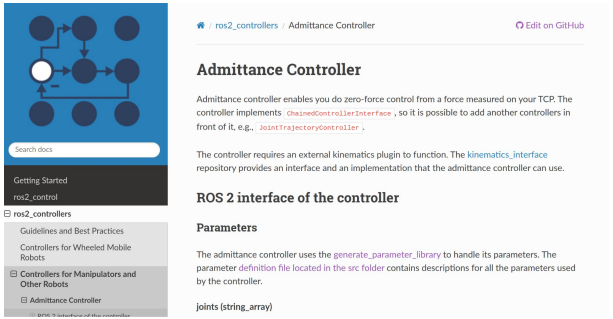


Regler mit ROS 2

Admittance Controller - *ros2_controllers*

- Standard controller für ROS 2 durch Picknik
- Schnittstellen:
  - Position ▲
  - Velocity 🔴
- Kraft-Momentensensor notwendig
- Trajektorienausführung (MoveIt)
- Werkzeugkompensation
- Eingebauter exponentieller Filter
- Aktualisierung der Steifigkeit

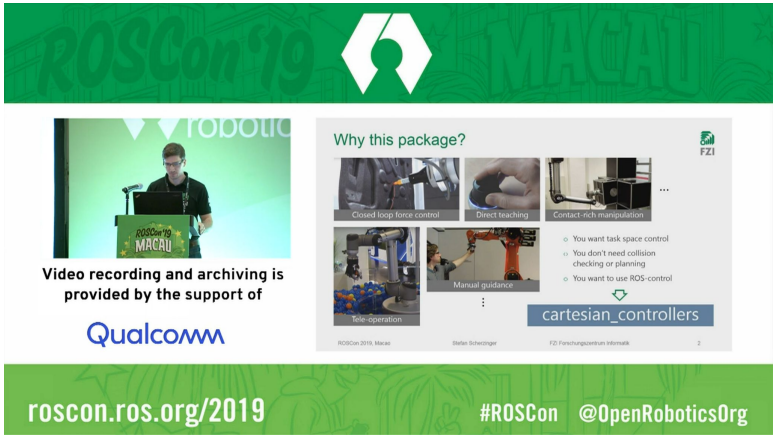




cartesian_controllers - FZI

- Das Kraftregelungspaket für ROS 1 (& ROS 2)
- Schnittstellen:
  - Position ▲
  - (Velocity 🔴)

- Kraft-Momentensensor notwendig
- Dynamisch anpassbar
- Werkzeugkompensation
- Oszilliert und vibriert gerne



Joint Impedance Controller - *iiwa_ros2*

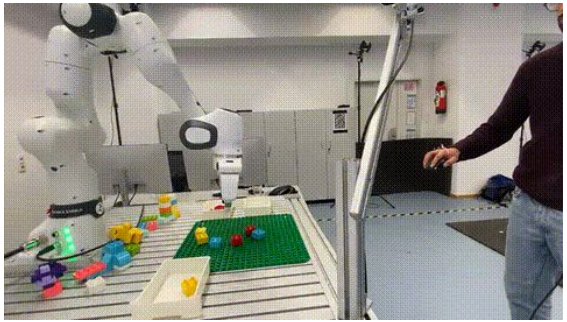
- Gelenke verhalten sich wie Feder-Dämpfer Systeme
- Schnittstellen:
  - Effort - Gelenkdrehmomente
- Kein Kraft-Momentensensor notwendig
- In *iiwa_ros2*, aber generisch geschrieben

CRISP - *Learning Systems and Robotics Lab (TUM)*

- Cartesian Impedance Controller
- Schnittstellen:
  - Effort - Gelenkdrehmomente ➜
- Fokus auf verstärkendes Lernen
- Gym-Integration
- Python bindings
- Beispiele für verschiedene Roboter

CRISP - Compliant ROS2 Controllers for Learning-Based Manipulation Policies

Authors: Daniel San Jose Pro[1], Oliver Hausdörfer[1], Ralf Römer[1], Maximilian Dösch[1], Martin Schuck [1] and Angela Schoellig [1].



ros2_effort_controller - *IDRA, University Trento*

- Cartesian impedance Controller
- Joint Impedance Controller
- Schnittstellen:
  - Effort - Gelenkdrehmomente ➜

- Nicht viel Dokumentation
- [Beispielrepository für KUKA iiwas]

README Apache-2.0 license

ROS2 Effort Controller

	Humble	Jazzy	Rolling
Branch main	Humble CI passing	Jazzy CI passing	Rolling CI passing
Branch kuka-prop-ctrl	Humble prop-ctrl CI passing	Jazzy prop-ctrl CI passing	Rolling prop-ctrl CI passing

This repository aim to create a robot independent torque controller based on ROS2 controllers. There is a base controller that communicate with the hardware interface of the robot, on top of that controller different types of controllers can be implemented. For now a cartesian impedance controller, a joint impedance controller and a gravity compensation are implemented.

Home

Davide Nardi edited this page on Jul 9 · 26 revisions

kuka_lbr_control Wiki

Welcome to the kuka_lbr_control wiki!
Here you can find the documentation on how to achieve **kinematics** and **impedance control** for a KUKA LBR iiwa 7, iiwa 14, med 7 and med 14 robots. The controllers are implemented according to the `ros2-control` real-time framework exploiting the state and command interfaces exposed by the lbr-stack repository which implements `ros2-control` the hardware interfaces for the lbr robots.

📝 Requirements

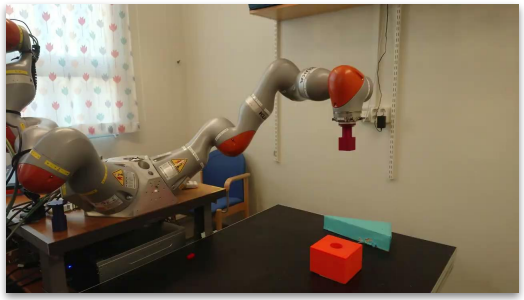
- ROS2 Humble or Jazzy
- ROS2 control: `sudo apt-get install ros-humble-ros2-control # or sudo apt-get install ros-jazzy-ros2-control`
- KUKA Sunrise app with Fast Robot Interface (FRI) support

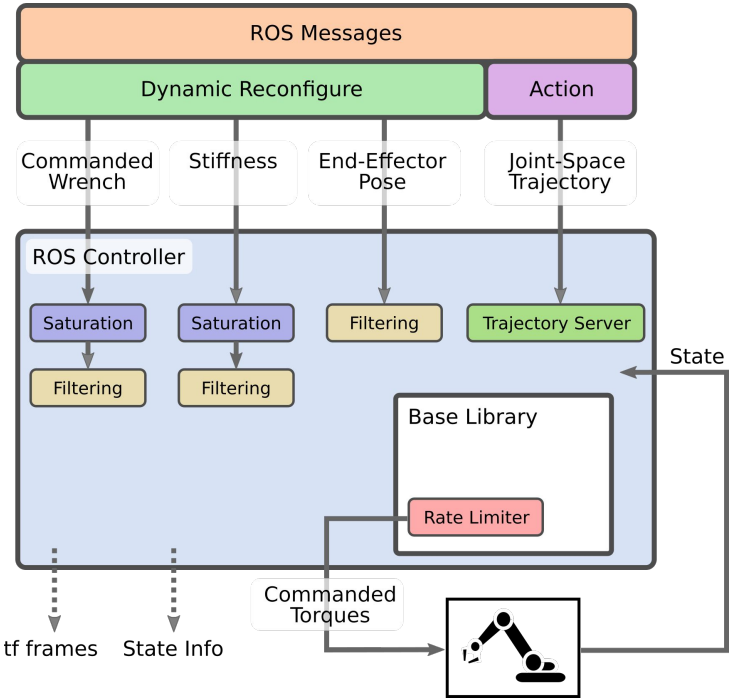
⚙️ Supported Control Modes

We currently support two control modes for the KUKA robot. Depending on the task you want to achieve, you can choose which version of our controllers to build.

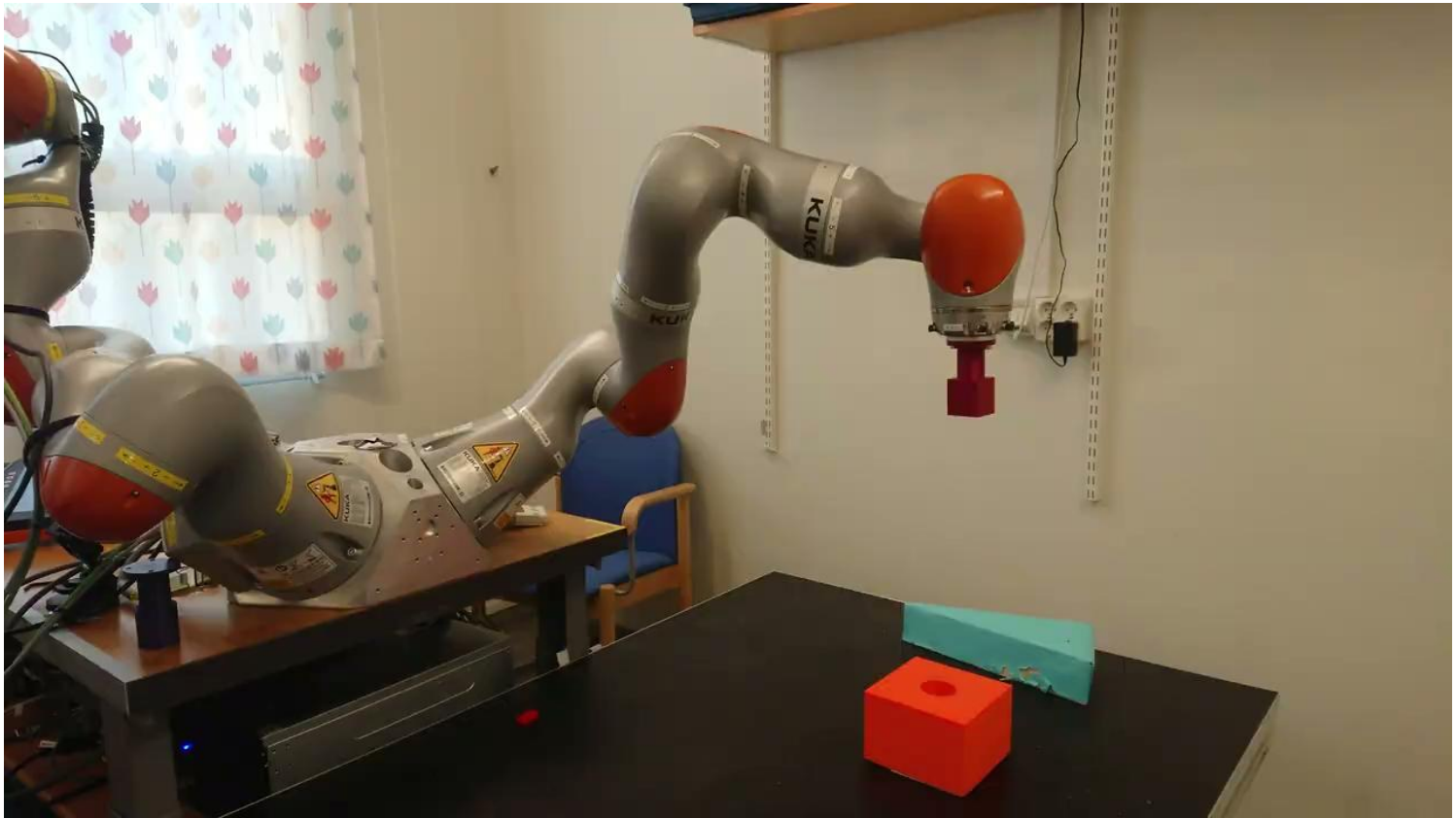
Cartesian Impedance Controller - Matthias Mayr

- Regler für ROS 1 & ROS 2
- Schnittstellen:
  - Effort - Gelenkdrehmomente ➜
- Läuft auf Frankas & KUKA iiwas
- Joint-Trajectory Ausführung (MoveIt)
- Dynamisch anpassbar

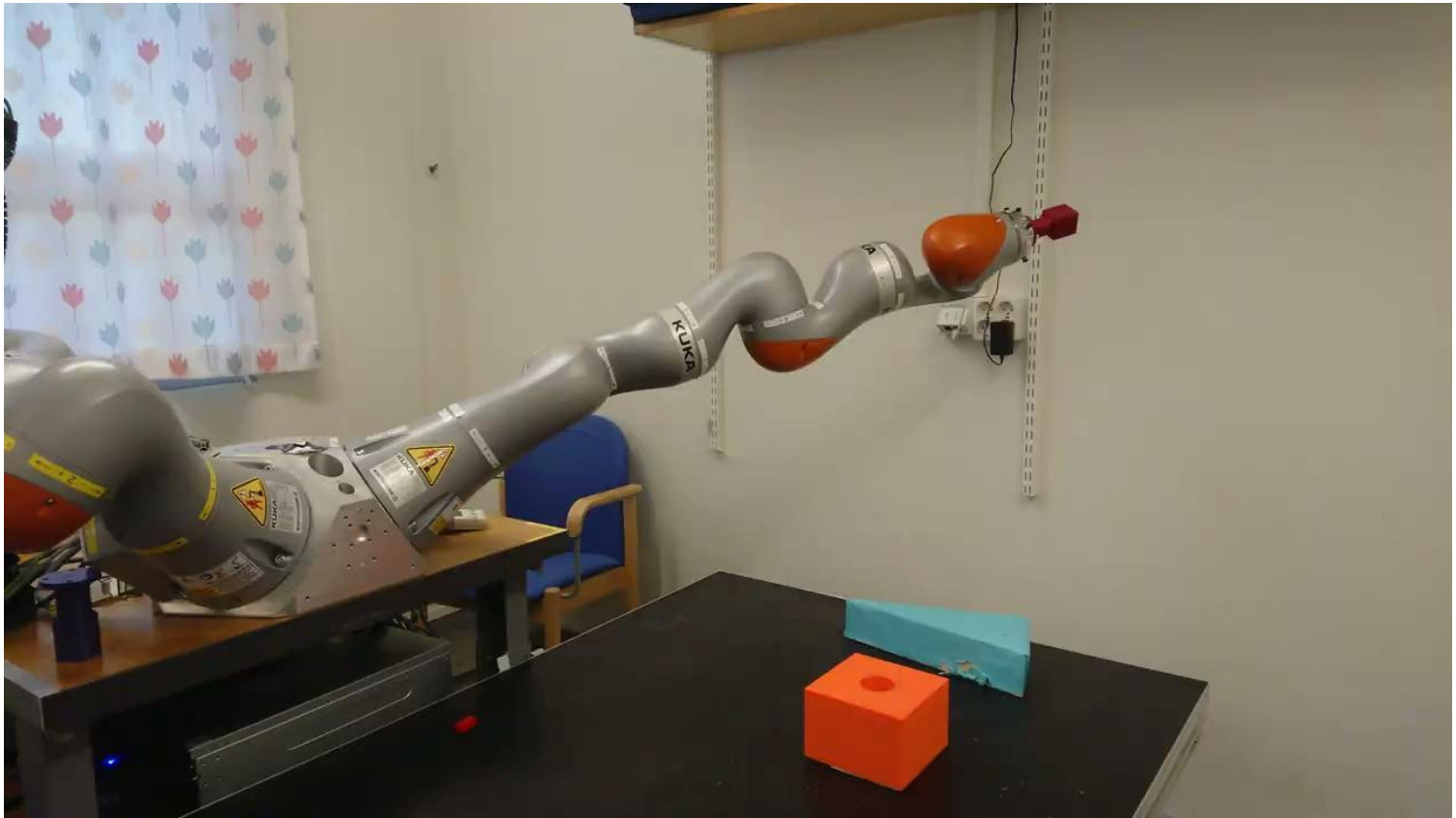




Für redundante Roboterarme



Stabilität in Singularitäten



Features

- Dynamic adaption of
  - Reference pose
  - Cartesian stiffness
  - Nullspace control
  - Commanded wrench
- Robot-agnostic implementation
- Base library
  - Integration into non-ROS software
- ROS Control implementation
  - Configurable with ROS msgs
  - Run (MoveIt) trajectories
  - Dynamic reconfigure integration

Joint work with Julian Salt Ducaju, Oussama Chouman

	KUKA FRI	libfranka	franka_ ros	**This project**
Cart. stiffness update	x	x	✓	✓
Wrench update	x	x	x	✓
Nullspace control	?	x	✓	✓
Kinesthetic Teaching	(✓)	(✓)	✓	✓
Trajectory execution	x	x	x	✓
Multi-robot support	x	x	x	✓

Regler für Roboter

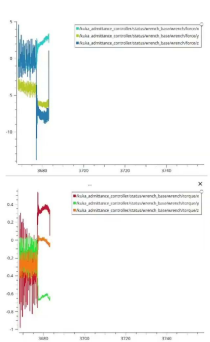
KUKA

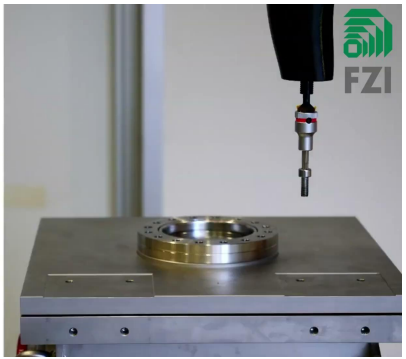
iiwa - Sunrise Cabinet

- **Treiber:** *lbr_fri_ros2_stack* https://github.com/lbr-stack/lbr_fri_ros2_stack
- **Schnittstellen:**
  - Position ▲
  - **Effort** ➜
  - (Wrench)
- **Regler:**
  - (cartesian_controllers)
  - Admittance Controller
  - **Cartesian Impedance Controller**

Industrielle Roboter - KR Controller

- **Treiber:** kuka_rsi_driver https://github.com/fzi-forschungszentrum-informatik/kuka_rsi_driver
- **Schnittstelle**: Position ▲
- **Regler:**
  - (cartesian_controllers)
  - **Admittance Controller**





Yaskawa

- **Treiber:** *MotoROS2*
  https://github.com/Yaskawa-Global/motoros2
  - **Hardware interface:**
    https://github.com/b-robotized/motoros2_hw_interfaces
- **Schnittstellen**:
  - Position ▲
- **Regler:**
  - (cartesian_controllers)
  - **Admittance Controller**

Franka Robotics

- **Treiber:** *franka_ros2* für *FR3*
  https://github.com/frankarobotics/franka_ros2
- **Schnittstellen**:
  - Position ▲
  - Velocity 🔴
  - **Effort** ➜
- **Regler:**
  - (cartesian_controllers)
  - Admittance Controller
  - **Cartesian Impedance Controller**

Universal Robots

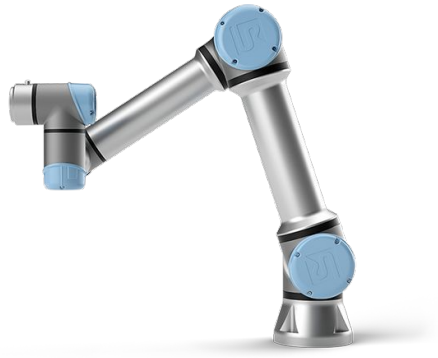
- **Treiber:** *Universal_Robots_ROS2_Driver* https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver
- **E-Series Schnittstellen:**
  - Position 🟢
  - Velocity 🔺
  - Effort ➜ **neu!**
  - Trajectories
- **Regler:**
  - (cartesian_controllers)
  - Admittance Controller
  - Cartesian Impedance Controller?

“Force Mode”:

- Einzelne kartesische Achsen kraftgeregelt
- Unter ROS nicht einfach kombinierbar

Joint Impedance

- URScript - bald mit neuer Schnitstelle (wip)



Zusammenfassung

- Kraftregelung für kontaktreiche Aufgaben
- Übliche ROS 2 Regler vorgestellt

<u>Anwendung mit Roboter:</u>

1. Verfügbare Kommandoschnittstellen
2. ROS 2 Treiberauswahl
3. Reglerwahl & ros2_control Konfiguration
4. Tuning

{Talk Repo}:

https://github.com/matthias-mayr/ROS2-Compliant-Arm-Control

{Code}:

https://github.com/matthias-mayr/Cartesian-Impedance-Controller