ISBN 978-82-326-3822-2 (printed version)
ISBN 978-82-326-3823-9 (electronic version)
ISSN 1503-8181

NTNU
Norwegian University of
Science and Technology

Mathias Hauan Arbo

Doctoral Thesis

Doctoral theses at NTNU, 2019:112

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Doctoral theses at NTNU, 2019:112

Mathias Hauan Arbo

# On Robotic Assembly and Optimization-Based Control of Industrial Manipulators

NTNU
Norwegian University of
Science and Technology

Mathias Hauan Arbo

# On Robotic Assembly and Optimization-Based Control of Industrial Manipulators

Thesis for the degree of Philosophiae Doctor

Trondheim, April 2019

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

**NTNU**
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-3822-2 (printed version)
ISBN 978-82-326-3823-9 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2019:112

Printed by Skipnes Kommunikasjon as

# Summary

Robotic manipulators are essential tools of industry. They are ubiquituous in modern factories, and they are starting to become more commonplace everywhere else. The trend in robotic manipulators is to move towards allowing for more human-robot collaboration, to allow for sensor-based tasks, and to perform classical industrial tasks such as assembly in new and unforeseen environments. These trends enable small and medium businesses to lower the cost of automation by simplifying robot programming.

The core interest of this thesis is an idealized automatic assembly system, capable of generating and executing robot programs based on CAD models. Such a system is challenging to realize, as it is composed of many subcomponents that are large research topics in and of themselves. From the control-theoretic perspective, much of the work lies in task specification and relating assembly of the parts to robot motions, as is the core discussion in one of the papers.

The papers present control strategies intended for on-line control of an industrial manipulator. As optimization techniques may have slow execution time, the timings of the different control strategies are a recurring topic in the papers. The communication and tracking latency of a KUKA robot system is also investigated when developing an open-source control interface to the KUKA system.

This thesis presents contributions in robotic assembly and optimization-based control of industrial manipulators. The papers in this thesis aim to address some shortcomings in the robotics literature that were found when investigating the idealized system. The accompanying text gives a brief introduction to the history of robotics, as well as an overview of some of the robotics literature.

# Contents

# Chapter 1

# Preface

This thesis is the result of my doctoral study at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) under the supervision of Professor Jan Tommy Gravdahl (NTNU), and co-supervisor Esten Ingar Grøtli, Ph.D., (SINTEF DIGITAL). The doctoral study was a part of SFI Manufacturing[1] and the project was funded by the Research Council of Norway under contract number 237900.

SFI Manufacturing is a cross-disciplinary center for research-based innovation[2] focused on competitive, high-value manufacturing in Norway. The center approaches this with three general research areas: multi-material products and manufacturing processes, robust and flexible automation, and sustainable and innovative organizations. The doctoral study was a part of the robust and flexible automation area.

During the doctoral study, I have had the pleasure of being co-supervisor to the master students: Ivar Eriksen, Lill Maria Gjerde Johannessen, Morten Andre Astad, and Shahrukh Khan. I have also had a three month research stay at the robotics research group of Production Engineering Machine Design and Automation division of Katholieke Universiteit Leuven. During the research stay I collaborated with Yudha P. Pane, under the supervision of Erwin Aertbeliën, Ph.D.

This thesis contains a collection of articles from my research as well as a presentation of theory to tie the research together. The thesis is an attempt at mastication of the hard topic of control for robotic assembly such that certain ideas and thoughts can be digested more easily. Robotics is a vast, discovered landscape of crooks and crevices, nooks and crannies, with pockets of ideas and implementations through which this thesis walks.

[1] `http://www.sfimanufacturing.no/`

[2] *Senter for Forskningsbasert Innovasjon* in Norwegian.

## 1.1 Acknowledgement

Three and a half years of a Ph.D. never pass in the blink of an eye; but thanks to the support and encouragement of my friends, family, and fellow colleagues, it has been a blast.

First and foremost, I wish to thank my supervisor, Jan Tommy Gravdahl, both for this opportunity, and for our many discussions. His support has been fundamental in getting through this. Secondly, I thank my co-supervisor and contact to SINTEF, Esten Ingar Grøtli, who has been eager to help, and taken an active role in my studies. My research would not exist if it were not for the support of SFI Manufacturing, and the Research Council of Norway.

I consider it an honor to have had a research stay at Katholieke Universiteit Leuven, and to have worked with Yudha P. Pane, Erwin Aertbeliën, and Wilm Decré. Their insight and discussions exponentially increased my robotics literacy. I am also grateful of Filippo Sanfilippo's contributions and discussions.

My Master students, both those who have finished and those who are soon to finish, have invested many hours in the complicated projects we have concocted. I am grateful for their many contributions to my work, both explicitly from software development and hardware setup, and implicitly from interesting discussions.

The thorough comments on my thesis by Jabir Ali Ouassou, Ingerid Brænne Arbo, Øystein Wigum Arbo, and my father Peter Arbo, were essential in ensuring its completion.

I am sincerely grateful for the wednesday presentation group and my lunch compatriots, who have kept me sane and satiated during both the long and short days. I would particularly like to thank my classmates who joined me in the step from Master student to Ph.D. student: Bjørn-Olav Holtung Eriksen, Erik Wilthil, Andreas Lindahl Flåten, Kristoffer Gryte, Håkon Hagen Helgesen. I am forever grateful for the complementarity and similarity of me and my office mate Bjørn-Olav, our companionship has been a cornerstone of my time at the department.

The last winter months would have been dark and cold if not for Carina Norvik, whose love and support has eased the burden of the thesis. Lastly, I would like to thank my family for all their support: my parents for allowing me to stray from the beaten path, my sisters for leading with example, and my grandmother for her strength and delicious dinners.

## 1.2 Contributions

The doctoral research includes contributions in:

- a globally exponentially stable speed observer for mechanical systems,
- model predictive path-following control for industrial manipulators,
- constraint-based robot programming using CAD information,
- novel controller designs for constraint-based robot programming with task-priority closed-loop inverse kinematics,
- and interfacing of KUKA robots with ROS.

The work is documented in the following articles:

1. M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "On the globally exponentially convergent immersion and invariance speed observer for mechanical systems," American Control Conference (ACC), Seattle, WA, 2017, pp. 3294-3299 [1],
2. M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "On model predictive path following and trajectory tracking for industrial robots," 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, 2017, pp. 100-105 [2],
3. M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "Mid-Level MPC and 6 DOF output path following for robotic manipulators," IEEE Conference on Control Technology and Applications (CCTA), Mauna Lani, HI, 2017, pp. 450-456 [3],
4. M. H. Arbo, Y. P. Pane, E. Aertbeliën, and Wilm Decré, "A System Architecture for Constraint-Based Robotic Assembly with CAD Information," IEEE International Conference on Automation Science and Engineering (CASE), Munich, 2018, pp. 690-696 [4],
5. M. H. Arbo and J. T. Gravdahl, "Stability of the Tracking Problem with Task-Priority Inverse Kinematics," IFAC Symposium on Robotics and Control (SYROCO), Budapest, 2018, pp. 121-125 [5],
6. M. H. Arbo, I. Eriksen, F. Sanfilippo, and J. T. Gravdahl, "Interfacing KUKA Industrial Robots with ROS for Research and Education," submitted to Elsevier journal of Mechatronics January 2019, and
7. M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "CASCLIK: CasADi-Based Closed-Loop Inverse Kinematics," submitted to IEEE journal of Transactions on Robotics (T-RO) January 2019.

The work has resulted in the following free and open-source software:

**CASCLIK [6]:** a Python module for prototyping of closed-loop inverse kinematic controllers for multiple constraint-based tasks,

**urdf2casadi [7]:** a Python module for converting unified robot description files or Denavit-Hartenberg parameters into symbolic functions for CasADi [8],

**Arbench [9]:** a FreeCAD [10] workbench for annotation of geometric features for robotic assembly,

**kuka_kvp_hw_interface [11]:** an alternative ROS hardware interface for control of KUKA industrial manipulators, primarily developed by Ivar Eriksen.

Other results during the doctoral studies not related to the thesis are:

a) H. H. Johnsen (artist), M. H. Arbo (technician), “Luminosity” (fiberoptic art installation), Galleri Nord-Norge, 2016,

b) M. H. Arbo, T. Utstumo, E. F. Brekke, J. T. Gravdahl, “Unscented Multi-Point Smoother for Fusion of Delayed Displacement Measurements: Application to Agricultural Robots” Modeling, Identification and Control, vol. 38, no.1, 2017, pp. 1-9 [12],

c) A. Busch, M. H. Arbo, S. A. Kasimba, K. Sripada, “PhD working conditions at NTNU with a special focus on PhD candidates with kids” The Interest Organization for Doctoral Candidates at NTNU (DION), 2017, and

d) H. H. Johnsen (artist), M. H. Arbo (technician), permanent fiberoptic art installation at Jessheim Videregående Skole, 2017.

Fig. 1.1 illustrates how the different papers are related. Paper 1 was an initial offshoot into the realm of nonlinear control theory before a course correction led to more industrially applicable research. The model predictive path following controller was first created for a two-link manipulator (Paper 2), and later extended for use with a UR5 and a UR3 (Paper 3). The model predictive approach was necessary for one of the controller formulations in Paper 7. The dashed lines indicate that the work with Paper 4 and Paper 6 are related to Paper 7. More specifically, the architecture and interface created in those papers are potentially useful with the library created in Paper 7. Paper 5 was a necessary step in furthering the theory used for one of the controller formulations in Paper 7.

Paper 4 won a joint best student paper award at the CASE conference and was the result of the three-month research stay at Katholieke Universiteit Leuven.

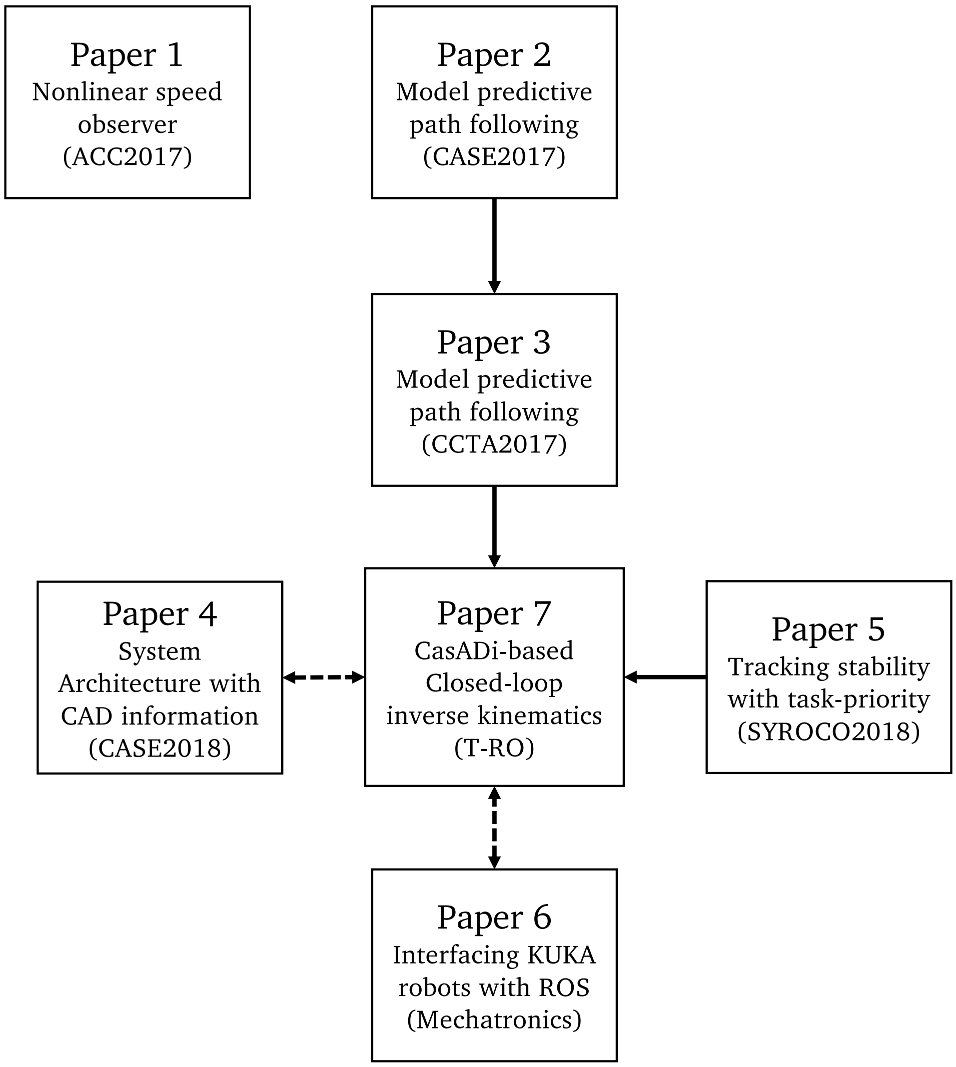

Paper 1
Nonlinear speed observer (ACC2017)
Paper 2
Model predictive path following (CASE2017)
Paper 3
Model predictive path following (CCTA2017)
Paper 4
System Architecture with CAD information (CASE2018)
Paper 7
CasADi-based Closed-loop inverse kinematics (T-RO)
Paper 5
Tracking stability with task-priority (SYROCO2018)
Paper 6
Interfacing KUKA robots with ROS (Mechatronics)

Figure 1.1: Relation between the papers.

# Chapter 2

# History and Motivation

“ *My story is a lot like yours, only more interesting ’cause it involves robots.* ”

Bender, *Futurama – 30% Iron Chef [13]*, 2002

## 2.1 History of Robotics

As much of modern technology, robotics has its beginnings in myths, legends, and science fiction. The term *robot*, from “robota”, meaning forced laborer, was introduced to the world in the play *Rossumovi Univerzální Roboti* written by the Czech author Karel Čapek in 1920. The robots were artificially created biological servants, echoes of earlier folklore and fiction such as the the myth of *Galatea*, the *Golem of Prague*, Leonardo Da Vinci’s mechanical knight, and many more. Already by 1928, the term robot had become commonly associated with these automata when “Eric the Robot”, a mechanical knight, opened the exhibition of the Society of Model Engineers in London[1].

In 1939, Isaac Asimov approached the idea of these mechanical men from a different perspective. Instead of fearing them, or using them to “point a moral” [14], he considered them as amicable or dangerous as any other equipment. In his own words: “They were simply well-designed machines” [15]. He coined the term robotics and roboticist for the study and students of robots as well as the three rules of robotics — the guidelines to minimize the harm robots can do to humans. Asimov’s work often explored the interplay between man and machine, taking the degrees of prediction and understanding shared between the two to new extremes. These texts would later inspire a young engineer named Joseph Engelberger.

As the story goes, in 1956 Joseph Engelberger met George C. Devol at a cocktail party [15]; an inventor who two years prior had filed a patent for the *Programmed Article Transfer* machine [16]. In cooperation, they realized the industrial potential of automated manipulators and formed Unimation Incorporated. Shortly thereafter, the Unimate (Fig. 2.1), the first industrial robot, became commercially available in 1961. Installed at General Motors in Trenton, NJ, it moved red-hot door handles and other die-cast parts to a pool for quenching without pain or problems for years. It was simply a well-designed machine. Robotics has since then exploded in popularity, and entered a variety of fields other than industrial manufacturing. Robotics now includes domestic service robots such as the Roomba [17], underwater swimming manipulators such as the Eelume [18], Mars-exploring rovers such as Curiosity [19], surgical robots such as the DaVinci [20], and a plethora of industrial manipulators. Tab. 2.1 presents a timeline with a small subset of the many interesting breakthroughs that have occurred in these 64 years of robotics research. The timeline tries to focus on industrial manipulators relevant for manufacturing and assembly and some of the tools surrounding them.

[1]With R.U.R. printed on Eric’s chest, the reference to Karel Čapek’s play is unmistakeable.

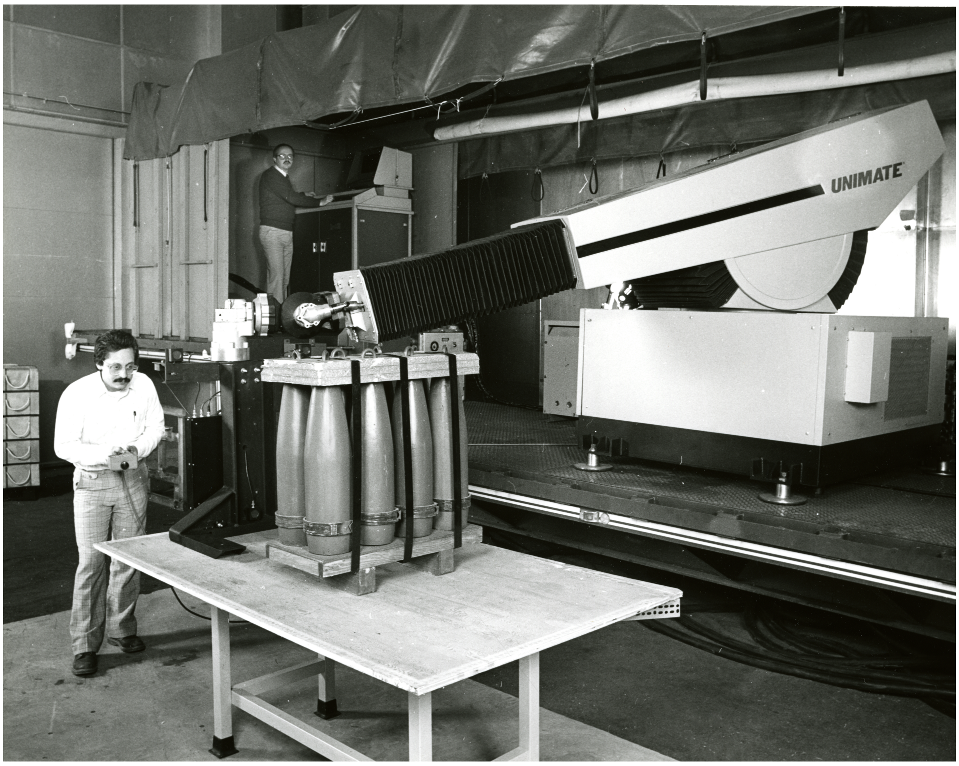


Figure 2.1: A Unimate at the Automated Manufacturing Research Facility (ca. 1983) being taught a new setpoint. Image courtesy of the National Institute of Standards and Technology Digital Collections, Gaithersburg, MD 20899 [21].

The first Unimate was a heavy, pneumatically-driven manipulator with 5 degrees of freedom (DOF). The earliest versions had bang-bang control, and programs could be taught to the robot by moving it to the desired positions which were recorded on a rotary memory drum [16]. The Unimate would sequentially move to the desired positions, allowing it to execute complex motions. In 1966, a Norwegian wheelbarrow company named Trallfa had difficulty with the labor-intensive and toxic spray-painting step in their production line. As a result of this, they decided to construct a spray-painting robot, one of the first exported programmable robots [22]. Robotics has been a turbulent market, with few robot manufacturers and products surviving throughout the times. ABB's IRB 6 (1974) was not only the first microprocessor-controlled robot, but also stayed in production for 17 years [23]. Another important milestone is the development of additional tools such as the remote center of compliance [24]. One can think of it as attaching a little spring to the end-effector of the robot, which allows it to flex as our fingers do when we

Table 2.1: Timeline of selected events in the history of robotics

| | |
|---|---|
| **1954** | Unimate patent filed by George C. Devol |
| **1961** | First Unimate robot sold to General Motors (5 DOF) |
| **1965** | The 6 DOF parallel-link Stewart platform was publicized |
| **1966** | Trallfa produces the first spray-painting robot in Norway |
| **1969** | Stanford arm, a 6 DOF electrically driven arm with force/torque sensing was created |
| **1973** | KUKA's 6 DOF FAMULUS robot enters the market |
| **1974** | ABB's IRB 6, the first microprocessor-controlled robot, is introduced |
| **1976** | Remote center of compliance device patent filed by Paul C. Watson |
| **1978** | The first SCARA robot prototype was developed by Hiroshi Makino |
| **1981** | Automatix introduces the first industrial robot with built-in machine vision |
| **1981** | The first direct drive arm was designed by Takeo Kanade |
| **1984** | Variable Assembly Language (VAL) II is introduced |
| **1985** | First 4 DOF Delta robot patent filed by Reymond Clavel |
| **1995** | The first light-weight robot (LWR) is created at DLR |
| **2002** | OROCOS is introduced as a flexible robot programming framework by Herman Bruyninckx |
| **2007** | The Robot Operating System (ROS) is created by Willow Garage |
| **2008** | UR5 and KUKA LBR 4 are the first commercially available cobots |
| **2012** | Rethink Robotics' dual arm Baxter enters the market |
| **2015** | ABB's dual armed Yumi enters the market |

hit a surface, complying to the external forces acting on the end-effector. This was an important breakthrough as it allowed robots to perform peg-in-hole insertions. It is an example of the many tools that were developed both before and after that can be attached to a robot platform to increase its versatility.

As offline robot programming and more complex behavior was desired from the users, robot programming languages became an important topic in robotics. The first generation of robot programming languages handled simple learning of poses and motions, and executed these in the prescribed sequence. They left robot systems "deaf, dumb, and blind" as Charles A. Rosen put it in the first *Handbook of Industrial Robots* [14]. As computer processors were added to the robot controllers, it became possible to formulate instruction sets for the robot. These formed the second-generation robot programming languages, of which VAL II [25] is a notable example. VAL II included functionality for offline programming, network communication, sensor integration, and real-time path modification. One can consider the second generation of robot programming languages to bring the robots senses and speech, albeit in a rudimentary form. These advances were possible as the instructions were handled by a microprocessor that was separate from the servo controllers. Each joint had a separate microcontroller that handled servo control, and the main microprocessor transmitted commands to the servo controllers. This resulted in a cascaded control hierarchy.

Most of modern research on control of robotic manipulators use systems that allow for control over network communication. The user is allowed to use their own hardware and software to control the robots. OROCOS [26], [27] and ROS [28], [29] are open-source frameworks that sprung up from this around the turn of the century, and greatly simplified the integration of existing works into new projects, as well as the validation and evaluation of others' work. The two frameworks may be used in conjunction with each other, and both are still in active use with ROS being the more popular platform. In May 2015, 9 million ROS packages were downloaded by over 70000 unique IP addresses [30], and the community of ROS users continues to grow.

Another important highlight was the development of light-weight collaborative robots that came with the 7 DOF KUKA LBR [31] and the 6 DOF robots of Universal Robots [32] in 2008. Before them, 6 DOF industrial manipulators were generally heavy, rigid machinery that required a large safety zone. The reduced inertia of light-weight robots meant that they could work in collaboration with humans without harming them. With advanced force-control techniques and backdrivability, an operator can manipulate them into desired poses by hand, quickly and easily. The additional joint on the KUKA LBR increases manipulability, allowing it to achieve a desired pose while still having an exploitable DOF.

With the Baxter [34] and ABB's Yumi [35] (see Fig. 2.2), from 2012 and 2015 respectively, the overlap between collaborative robots, humanoid robots, and service robots shifted. The dual-arm design allowed for easier handling and manipulation of objects in an intuitive way for unskilled workers. And although Baxter was well received when it came out, the company producing them, Rethink Robotics, was another victim of the turbulent robotics market. Rethink Robotics went out of business in October 2018, following a lack of market success.

Although robotics is a relatively young field, the ingenuity and innovation of its many inventors and researchers has made it a flourishing industry with

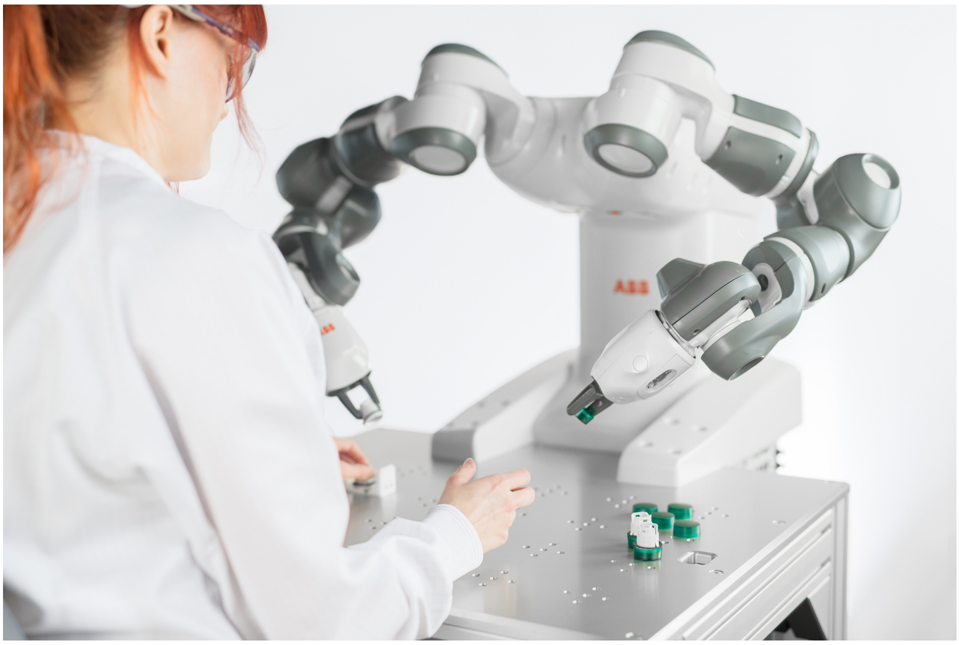


Figure 2.2: The collaborative, dual-armed Yumi. Image courtesy of ABB [33]. Photo: ABB.

multiple competing companies. In 2015, the worldwide robotics spending was $71 billion, and projected to reach $135 billion in 2019 [36]. At the risk of projecting my own desires and expectations onto the timeline, the research seems to move from humanless factories to collaborative robots, from dedicated programming languages and computers to control over network and open interfaces, from a single form factor and set number of DOF to multiple DOF and methods of mechanical linkage. These trends spell out an opportunity for robots to be more customizable, potentially even modular, for robot skills to be transferrable, and for the advancement of collaboration between man and machine.

## 2.2 History of Automated Assembly

In 2015, programming and installation constituted 35% of the total cost of a spot-welding robot according to Boston Consulting Group [37]. By 2020, the portion is projected to constitute 33% of the total system cost. Although the numbers do not describe assembly processes, it is reasonable to assume that systems engineering for a set of assembly processes would exhibit a similar or greater percentage as spot-welding generally involves point-to-point motion whereas assembly requires both trajectory tracking and potentially force-controlled motions. For high-volume product series, the relative cost of programming is low with respect to the potential profit. For low volume, individual customization of parts, or agile assembly scenarios, the cost of programming is a significant portion of the total cost of automation — a portion that may make robotic automation out of reach for small to medium businesses.

In an ideal scenario, any design that could be dreamt up would be possible to construct with automatically programmed robots. A *fully automated assembly system* software would analyze the design, find potential pitfalls, generate assembly sequence plans, select and tune the robot skills, and program the appropriate robots. Fig. 2.3 illustrates a robot that has been automatically programmed based on a CAD model, about to execute the assembly. A complete version of such a software system does not yet exist, but some systems have been created that attempt to address the issue. The purpose of this section is to look at automatic robotic assembly for manufacturing on a system level. The thesis has no opportunity to address all aspects of such a system, but the work attempts to trace its shadow and cast light on some architectural and control choices that may be desirable.

A fully automated assembly system consists of three core parts: analysis, planning, and execution. Most assembly systems have been designed from the motion-planning perspective. To be a true fully automated assembly system, it must also consider the execution of the assembly plan. As finding two-handed assembly sequences for arbitrary designs is NP-hard [38], it is debatable whether a true fully automated assembly system is feasible or reasonable to expect, but they describe at their core an object-level robot programming. Investigating previous iterations of such systems gives insight into how one may better look at object-level, also called task-level, robot programming for assembly.

BUILD [39] from 1974 was one of the earliest model-based assembly planning systems. It was capable of stacking simple cubic or triangular blocks to a desired design. Coming from early artificial intelligence, it used heuristic searches and a simple *world model* to analyze which stacks to construct first,

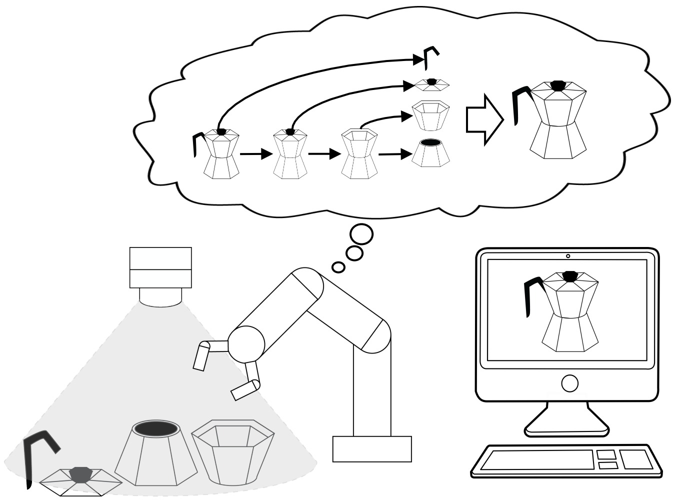


Figure 2.3: Automated assembly programming from CAD to real-life robots.

and whether the stacks would result in stable subassemblies. World models are a common theme in fully automated assembly systems and they are integral for model-based robot programming approaches. The world model is a representation of the world that is used during planning and control of the robot system. Limitations of the world model may inherently limit model-based control approaches. For instance, with a world model where link frames only have one parent frame, it will not be possible to represent the closed-loop kinematic chains formed in certain contact situations.

In 1977, Lieberman and Wesley presented AUTOPASS [40], an “AUTOmated Parts ASsembly System” that attempted to put automation into the realm of lower volume production with assembly-directed programming of robots. Here assembly-directed means that the user provided an assembly specification rather than manipulator motions when programming. AUTOPASS was a high-level programming system where English-like statements were used to define the assembly, and a world model maintained the relative transformations between parts. The statements, in combination with the world model, defined the manipulator-level motions that were required. AUTOPASS contained various modules to execute the desired assembly tasks, e.g. collision-free “pickup” trajectory generation, or collision-free “put-down” trajectory generation. The modules calculated trajectories that were executed by the robot, and the modules could account for sensor data to ensure that the assembly had been achieved. Other notable automated assembly systems from the period are LAMA [41] by Lozano-Perez (1977), RAPT [42] by Popplestone (1978), and TWAIN [43] by Lozano-Perez (1986).

In 1996, Kaufman et al. presented the *Archimedes 2* [44], a "mechanical assembly planning system" that allowed for CAD-level programming of a robot workcell. The system used a pre-STEP model of parts, and a set of files describing how the parts were joined, recommended subassemblies, and suggested assembly directions. The file describing the method of part joining was pertinent to the assembly planner as pressfits, snapfits, and threaded contacts would result in overlap of the parts even though there are feasible assembly directions. The system described three core assumptions: that all parts behave like rigid bodies, that mating is achieved by trajectories that are translations, rotations, or screw motions, and that the assembly could be performed by mating or joining of the parts for which the necessary tool was known. Mating was establishing a relative positioning of the part to be assembled following a certain trajectory type along an assembly direction. Joining was attaching the parts once in the correct relative positioning, e.g. to weld or glue. The system had two assembly planners: the first considered the part geometry and tool accessibility, and the second considered the assembly sequence, subassembly reorientation, and tool changing. The resulting sequence of actions were translated to Adept's V+ [45] robot programming language for execution. This means that the robot program was a joint position controlled sequence of poses and mating trajectories, but may have allowed for sensor data to ensure that the assembly had been achieved.

*HighLAP* [46], [47] by Thomas et al. was a "HIGH-Level Assembly Planning" system developed in 2001. The system[2] saw various improvements [48]–[50], and we will consider its features from the 2010 presentation in *Robotic Systems for Handling and Assembly* [51]. Thomas et al. presents a system architecture similar to Archimedes 2 in that it takes a STEP file with annotated part constraint information and uses a geometric engine to reason about the assembly plan and mating directions. The constraints are used to define the positioning of the objects, and the mating directions are used to infer the ordering of the assembly operations. The assembly sequence is described by a "net" of manipulation primitives. A manipulation primitive describes a continuous control mode of the manipulator, e.g. constant rotation around a task frame. The "net" of manipulation primitives is a finite state machine that switches between manipulation primitives based on sensor information.

Recurring aspects in these systems are recurring problems in robotics. In 1983, Lozano-Perez, the creator of LAMA and TWAIN, summarized five core requirements of robot programming [52]: sensing, world modeling, motion

[2]For simplicity, the works by Thomas et al. are referred to as HighLAP, but it may be a misnomer. The later articles of the research group do not explicitly state that the work is a part of a specific software solution called HighLAP.

specification, flow of control, and programming support. The specification and standardization of these requirements have been the topic of many research groups, and we see that they are echoed in the history of automated assembly.

The systems presented generally consider CAD files annotated with auxiliary assembly information. This information contains geometric constraints between part features, such as with RAPT, Archimedes 2, and HighLAP, and information about necessary tools or type of assembly task to perform. To the author's knowledge, the first use of such features for positioning or describing the relative motion of the parts during assembly is from 1975 by Ambler and Popplestone [53]. A formal part-mating model for a variety of geometric primitives was described in 1990 by Kim [54]. It included compliant motion to compensate for part, environmental, or positioning uncertainty. An example of features in a screw insertion task is shown in Fig. 2.4

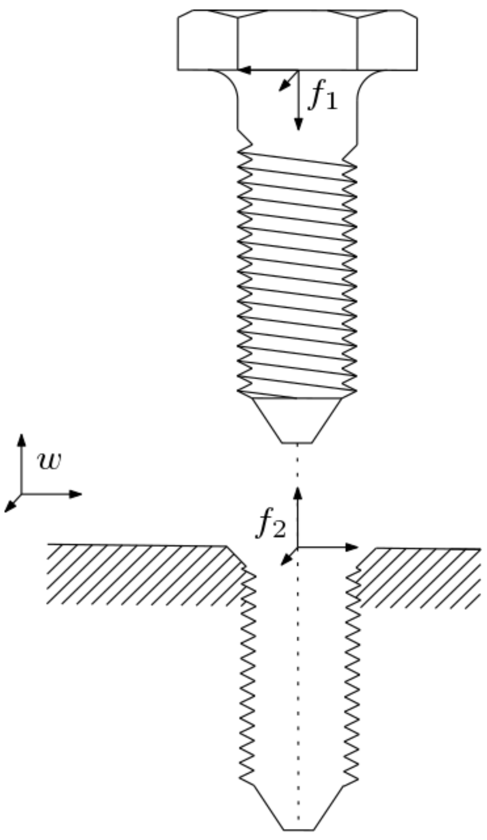


Figure 2.4: Visualization of constraints between geometric features. The screw is described by a reference frame $f_1$, and its hole by $f_2$, the hole and screw are aligned by aligning the axis of the two reference frames. The assembly direction is along the axis.

As inferring assembly sequences from the full CAD model has long been too computationally costly, representations of assembly sequences is another cornerstone of automated assembly systems. The *liaison graph* [55] originally developed by Bourjault (1984), further developed by Fazio et al. [56] (1987), describes assembly relations between parts as a graph with nodes representing components of the assembly, and edges referred to as *liaisons* describing an abstract assembly relation. The precedence graph is formed by querying the user on which edge must be established before the other until a description of the full set of assembly sequences is formed. The AND/OR graph [57] by Homem de Mello and Sanderson (1990) is a more succinct representation of all possible assembly sequences. The propagation of uncertainty, and informing the planner about essential paths in the liaison graph was introduced with the *key characteristics* [58] by Whitney (2004). This discussion barely scratches the surface of assembly representation and sequence planning but is intended to give some reference points for the reader.

Early automated assembly systems seem to have sprung up from early artificial intelligence. Heuristic searches, a cornerstone of artificial intelligence of the late 1970s, showed promising results in many robotics tasks, and it had an intuitive connection to assembly. Artificial intelligence is also a core component of model-free approaches in robotics, and has been applied to assembly from the work of Ikeuchi and Kang [59] (1990[3]) to the Horizon 2020 project *Smart Assembly Robot with Advanced Functionalities* [60]. As this is a large topic in and of itself, and most of the artificial intelligence approaches consider automated assembly from the production of the product stage rather than the design stage, model-free approaches are considered outside the scope of this thesis.

There have been many architectures proposed for the fully automated assembly system. One may wonder if the differences between the architectures are like a three-legged table versus a four-legged table. Neither of which is incorrect, it is a matter of design, users, and taste. The recurring core concepts that seem powerful are geometric constraints between parts, separation of part or CAD-level planner and the robot planner, sensor integration (particularly force sensing), and using a mid-level controller that translates the object-level tasks to commands that can be applied to the robot system.

[3]First reported by Carnegie Mellon University in either 1990 or 1993, depending on which subsequent sources one reads.

## 2.3 Why Research Automation?

> *They have no loads to carry: the machine carries the load. They have not to lift and push: the machine lifts and pushes. They have nothing else to do but eternally one and the same thing, each in this place, each at his machine.*
>
> Thea von Harbou, *Metropolis*, 1925

Automation has come, and automation is coming. Industrial robots are cheaper, more reliable, and often better than human workers at many repetitive tasks. Research in robotics has focused on allowing robots to gradually take over a greater range of tasks, with intelligence, versatility, and small-scale production in mind. Automation in its many forms, computerization and robotization, affects a broad range of industries, and to reflect on ones research should also include reflecting on its effect on society. This section considers some of the concerns voiced regarding technological unemployment, and attempts to present my view on the matter.

In the introduction to *Cybernetics* [61], Norbert Wiener states "the first industrial revolution, the revolution of the 'dark satanic mills,' was the devaluation of the human arm by the competition of machinery. [...] The modern industrial revolution is similarly bound to devalue the human brain, at least in its simpler and more routine decisions.". This quote states the idea that each industrial revolution, and technological advance in automation, brings with it a devaluing of a human attribute in the manufacturing process. Devaluing gives associations with a comparison of man and machine, where machines are developed with attributes comparable to a human worker. This then leads to the machine taking over the human's job, which is referred to as technological unemployment. If there is a limited number of jobs, this will cause problems. But human workers are trainable, and can find other occupations or create new ones. The problem occurs when it becomes difficult to know which areas are threatened. Many students nowadays fear that their education may become obsolete shortly after completion, or that whole occupations are threatened[4]. The fear of robots taking our jobs seems to have gained more traction in popular culture in recent years. Possibly as a result of the high media coverage of emergent technologies, as well as recent

[4]As evidenced by NRK's 2018 article announcing the nationwide university application deadline that was titled *Her er de tryggeste utdanningsvalgene* (translation: Here are the safest educational choices) [62].

reports that identify and quantify automation's effect on occupations. We will take a look at two of these methods of identification and quantification in particular.

In 2014, the Swedish Foundation for Strategic Research (SSF) presented a report titled *Vartannat jobb automatiseras inom 20 år*[5] [63]. The report was a Swedish version of *The Future of Employment* [64]: a report by Carl Benedikt Frey and Michael A. Osborne from 2013[6]. The technique considered the likelihood of jobs to be automated. In the report by SSF [63], this was adapted to the Swedish job market, giving the estimate of 53% of jobs being replaced by automation within 20 years.

The method of Frey and Osborne works by identifying professions in terms of whether there is something stopping them from being automated [64]. The bottlenecks of automation were grouped into three categories: perception and manipulation, creative intelligence, and social intelligence. These categories were further split into subcategories. In the article of Frey and Osborne, 70 occupations were manually classified on a scale from 0 to 1 describing how closely they related to each of the subcategories, and a classifier was trained on the data. The results of the manual classification were assumed to be a noise-corrupted observation of the true classification values. Given that their manual classification was sufficiently correct, and their categorization was correct, the trained classifier would give an accurate likelihood estimate. The classifier was then applied to 702 occupations in the O*NET occupational classification database [66]. Resulting in an estimate of 47% of the US occupations being in the "high risk category", jobs that Frey and Osborne refer to as being automated "relatively soon, perhaps over the next decade or two." Both Frey and Osborne and SSF note that the method only considers the technical bottlenecks of automation, not the economic ones. Another limitation of this method is that it assumes that occupations are directly described in terms of the bottlenecks, describing the likelihood of the occupation itself being automated rather than activities within the occupation.

The report *A Future That Works: Automation, Employment, and Productivity* from 2017 by McKinsey Global Institute [67] presents a more in-depth analysis of the effect of automation on employment and productivity growth. The key insight of the analysis is that activities in an occupation are not the occupation itself. The McKinsey analysis considers the likelihood that activities of an occupation will be automated. This perspective differs fundamentally in that it acknowledges that there will be a paradigm shift, but that the shift

[5]Translation: Every other job is automated within 20 years.

[6]The research results were made public in 2013 [64], but published in *Technological Forecasting and Social Change* in 2017 [65].

introduces changes that affect aspects of occupations.

The McKinsey report describes a framework for evaluating the automation potential of economies using 18 capabilities. The capabilities are divided into 5 main areas: sensory perception, cognitive capabilities, natural language processing, social and emotional capabilities, and physical capabilities (e.g. generating novel patterns is grouped as a cognitive automation capability). The automation capabilities are evaluated by whether the estimated level of performance of the current state of the art within the field is roughly below, equal, or above the current median human level. This is a more granular categorization and evaluation than used in the Frey and Osborne approach, which handles a larger dataset of activities, and spans most of the world economy. The McKinsey report also identifies factors affecting the pace and extent of automation: technical feasibility, cost of deployment, labor market dynamics, economic benefits, and regulatory and social acceptance. The report estimates that the employee-weighted overall percentage of activities that can be automated by adapting new technologies in Norway is 42% [68].

The result of these reports are daunting. They describe workers entering a roller coaster ride without handlebars. Where twists and turns may throw workers off, leaving them running to catch up with the future. Some of these concerns were addressed in *Second Machine Age* by Brynjolfsson and McAfee [69] from 2014. They described how digitization of information, exponential growth of computing power, and the combinatorial nature of innovation imposes difficulties on technological innovation and can lead to greater income inequality. They described how technology leads to a bounty of options and methods, but a spread of growing differences in income, wealth, and other circumstances of life. "Analog dollars are becoming digital pennies" [69] as services and products are distributed instantaneously over the internet and along other previously unforeseen avenues. In the field of information technology, whose trends Brynjolfsson and McAfee leverage to describe their predictions of the world economy, Marc Andreessen, co-founder of Netscape which produced the first widely used web browser, has stated that "software is eating the world" [70]. It has been expected that this also encompasses manufacturing.

In 2011 the German government started a project named *Industrie 4.0* [71] that promotes computerization and data exchange in manufacturing systems. They believe that we are on the cusp of the fourth industrial revolution, the era of cyber-physical systems, Internet of Things, and cognitive computing. Bahrin et al. describes it as "the next phase in a digitization of the manufacturing sector" [72]. The idea stems from observing the trends of information technology and its effect on neighboring industries. As this inevitably approaches manufacturing, we can be aware of the next industrial revolution before or as it is happening. The German government has approached this by bringing forth the discussion of Industrie 4.0 in an attempt to preemptively enact policies and foster projects to better handle the industrial revolution. Similarly, the Norwegian government started the project named *Digital21* [73] to promote digitization of Norwegian business sector. Whether these projects will primarily benefit corporations or secure workers is yet to be determined.

Knowing the potential pitfalls, why should we, as individuals, research automation? Jens Glad Balchen, the founder of the Department of Engineering Cybernetics, stated that "Ingen mennesker er tjent med slavearbeid" [7] in an interview with Aftenposten [74] in 1966. This mantra means that automation is the tool that removes tedious tasks and leaves the creative control in the hands of man. This is the idea of automation as labor substitution, gradually removing aspects of work which in turn may lead to technological unemployment. However, as described in the McKinsey report, automation and the current technological advances bring not only labor substitution, but augmentation and optimization of current activities. This allows workers to perform tasks with a higher productivity and accuracy than ever before as we push the boundaries of what work is, and how it is perceived. Both the *Second Machine Age* and the McKinsey report describe how automation has the opportunity to mitigate stagnating productivity gains. Productivity gains which may previously have been ensured by having a large working-age population with respect to the general population and sufficiently high birthrate [67]. They envision, and consider necessary for increasing productivity, a workforce where man and machine are in cooperation, where the creative and interpersonal skills become more important, or to quote the McKinsey report: "Automation could make us more human". This vision may present unique opportunities for skilled workers, but does not address the issue of the unskilled worker.

A trend in information technology rarely addressed in manufacturing and presentations of Industrie 4.0 is that of the open-source movement. In my opinion, an important enabler of the rapid development of information technology is the early open-source attitude[8]. This enabled the establishment of new companies by creating standardized interfaces and communication protocols with which new and hitherto unheard-of services and products could be provided. It also allowed new actors to enter the market without major initial investments in software infrastructure [76]. Similarly, robots are acquiring and distributing skills at an unprecedented rate thanks to open-

[7]Translation: No man benefits from slave labor.

[8]E.g. the free and open-source Apache HTTP Server from 1995 quickly became the leading HTTP server, and has remained so since [75].

source robotics projects such as ROS [28], [29], OROCOS [26], [27], and others. They allow us to quickly distribute our work, and incorporate existing methods into new approaches. Various free online tutorials are now bringing robotics to the younger generation with realistic dynamic simulations and levels of control. It is my belief that this has increased the innovation potential of robotics significantly and will be an important step to bring about the new industrial revolution.

Brynjolfsson and McAfee outline various potential approaches to handle the onslaught of technological development, of which one appears addressed to the unskilled worker and is of particular interest to researchers in robotics. For students and workers of the future they recommend to “fill up your toolkit and acquire skills and abilities that will be needed in the second machine age”. We, as researchers within the field, have the opportunity to design these toolkits. By working in education and making the tools and methods we design publically available we will democratize automation. This will lower the entrance cost, enabling the unskilled worker to become skilled. To quote Glaucon in Book II of the *Republic* by Plato: “the tools which would teach men their own use would be beyond price”. That is my motivation for researching automation. To make the esoteric art of robotics commonplace, to create tools which would teach men their own use.

# Chapter 3

# Robot Control

## 3.1 Introduction

Robots are the culmination of electronics, mathematics, and physics. They are the bridge between the digital and the physical. The interacting computed will. And most of the time they are simply a couple of motors, sensors, and microcontrollers hooked up to a computer.

Fig. 3.1 shows the overall control hierarchy considered in the thesis. Such robot control hierarchies are commonly split into low-level dynamic control, controlling the torque on the motors, and mid-level kinematic control, controlling the joint position and velocity. Recent commercially available robots such as the Franka Emika [77] and KUKA LBR IIWA [78] allow for low-level joint torque setpoints suggesting that this distinction may change. Describing the low-level controller as a dynamics controller also ignores any control of current or electrical characteristics of the servo-control loop. In this thesis, the kinematics–dynamics separation is sufficient for describing the work in the papers.

This chapter presents some of the literature on low-level and mid-level control to entice the curiosity of the reader, and place the papers in a larger context. The chapter ends with a brief discussion on the results and potential future work.

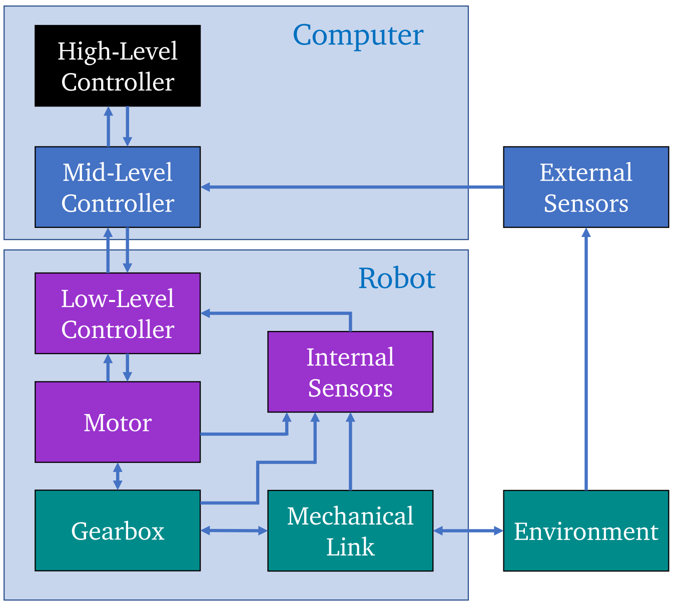


Figure 3.1: Abstract representation of the overall control hierarchy considered in the thesis.

## 3.2 Low-Level Controller

The low-level controller is concerned with actuator control [79]. Most robots are electrical, and a popular choice of motor for actuation is the permanent magnet synchronous motor [80]. For direct-driven robots [81], control of the motors directly relates to the joint angles. However, most modern robots have either gearing with low backlash such as harmonic drives [82], or gearboxes and belts to transmit the power from the motor to the rotational or prismatic joint on the manipulator [83]. Fig. 3.2 shows how the motors for the last three joints of the KUKA KR16-2 are on its shoulder, and the transmission goes down the length of its forearm. These transmissions are generally designed to have a high mechanical stiffness as the control of rigid manipulators is well established [84], but they have an inherent flexibility, friction, and may exhibit backlash. For an overview of the control of flexible joint robots, the survey paper by Ozgoli and Taghirad [83] is an excellent starting point. They state that the spring constant in industrial robots is very high, which means that the effect of the gearbox dynamics is likely to only adversely affect the robot when moving at high speeds or with heavy loads.

Joint angle, joint speed, or joint torque setpoints are common control modes described in the control literature. Most industrial robots are only equipped with a joint angle sensor which is usually placed on the motor side

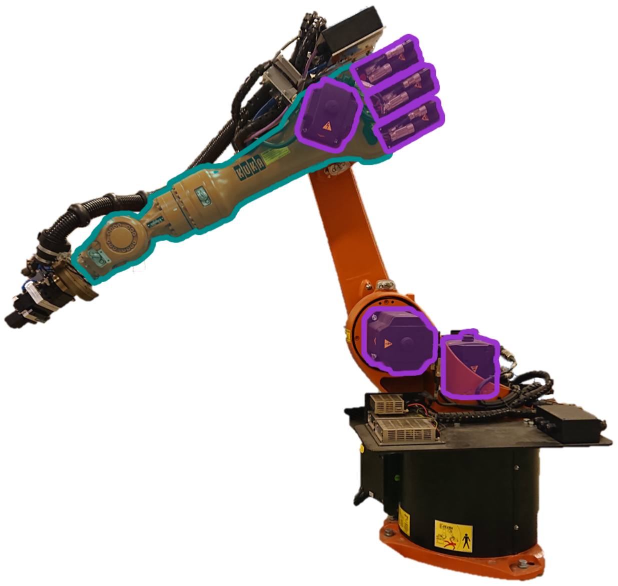


Figure 3.2: The KUKA KR16-2 industrial robot with motors highlighted in purple and the long transmission down link A3 highlighted in green.

of the transmission [85]. This means that in an ideal world, with perfect knowledge of the parts and their placements, and an infinitely fast motor controller, the transmission is still a source of uncertainty. A transmission that is subject to wear and tear, heat expansion, and other time varying effects [80]. Accurately modeling the dynamics of the robot system is therefore an important aspect of the low-level controller.

According to Hedberg et al. [86], a common scheme for control of industrial robotic manipulators is model-based feed-forward combined with *decentralized PID control* of the motor angles. Decentralized PID control refers to controlling each motor angle separately, while model-based feed-forward involves adding a feed-forward term based on the modeled dynamics. This is intended to make the control of each servo separable, and each motor has its own PID controller. The joint velocity is often found using a derivative filter. In his doctoral thesis [87], Andreas Stolt provides a block diagram of the low-level joint controller on the ABB IRC5. Descriptions of the low-level joint controllers are not readily available from most robot manufacturers. One can generally expect industrial manipulators to have motion buffers that introduce a tracking latency from when a signal is sent to the robot, until it is able to achieve the desired joint position [88].

For collaborative robots such as the UR5 or the KUKA LBR IIWA, the motors are situated in the joints with a harmonic drive transmission. The DLR lightweight robot, which served as the basis for the KUKA LBR, is documented in [82]. It has strain gauges placed in the flex spline of the harmonic drives to measure the torques on the links themselves. The UR5 does not provide a joint torque setpoint interface, but it does have a 250 Hz joint velocity interface. This was used in Paper 7 to implement compliance with respect to forces and torques.

The low-level controller may also allow for *operational space control* [89]. Operational space control refers to control of the end-effector in Cartesian coordinates as most tasks can be parameterized by the 6 DOF present in everyday life. This means that the low-level controller has setpoints for the pose or pose velocities of the end-effector. It may also allow for direct force control methods to describe the forces and torques of the robot in its environment [90] and hybrid force/position control strategies exist to control the robot in certain modes in certain directions [91]. For accurate control in such a scenario, environment modeling can be necessary [92] including the stiffness and damping of the workpiece. Operational space control means that the inverse kinematics are handled in the low-level controller. Delegating the inverse kinematics problem to the mid-level controller instead, and having a standardized interface to the low-level controller on robots may open up for flexible robot assembly programming architectures as described in Paper 4.

## 3.3 Mid-Level Controller

The mid-level controller is a translator and compensator. It translates the intent of the higher-level controller to something that can be sent to the low-level controller, and using modeling, learning, and sensors, it compensates for limitations in the robot system or its environment. The intent of the higher-level controller is a difficult topic and the separation of the layers of such a system is a recurring discussion.

In the *Industrial Robotics* chapter of the *Handbook of Robotics* [79], Hägele et al. (2008) describe different levels of programming of a robotic welding systems. As a thought experiment it may be interesting to apply the same terminology to an idealized automated assembly system.

- A *product*-centric system would be able to automatically generate feasible assembly sequences and process parameters such as insertion force, directly from the product design and material information. Then it would be deployable to the available robots and inform about manual tasks.

- A *process*-centric system would be able to take a description of the assembly processes required (e.g. peg-in-hole insertion) and its process parameters to automatically control the sensor-based motions of the end-effector.

- An *arm*-centric system would require explicitly defining end-effector paths and trajectories for the process in Cartesian space.

- A *joint*-centric system would require explicitly defining joint-space paths and trajectories.

Paper 4 tries to investigate a product-centric system architecture. It describes a high-level coordinator as an assembly planner and finite state machine that orchestrates transitions between continuous control modes of the robot. The mid-level controller is a constraint-based multiple-task controller [93] that gives joint velocity commands to the low-level controller on the robot.

When the mid-level controller is used to follow a desired path planned by a high-level controller, Kröger and Wahl refers to it as an *on-line trajectory generator* [94]. The *Iterative Learning Control* [95] method, where repetitive errors of a robot system are compensated for by modifying the control signals, can also be considered a mid-level controller as the higher-level intent is a repetitive path, and the control inputs to the low-level controller are modified. Model predictive path-following controllers such as the ones presented in Paper 2 and 3, based on the work of Faulwasser et al. [96], [97], are other examples of mid-level controllers where the intent is to follow a path while adhering to certain limitations.

We can see that this general concept of mid-level control described in the beginning of the chapter leaves us with an onion. At any point as we peel layers from our system until we hit the robot's communication interface, we could stop and call this the mid-level controller. And each previous layer presented a new intent to be translated. The general description is simply the interface from a conceptual intent to something that can be sent to the robot system. There appears to be no clear answer to the question of where the separation of the high-level system and the mid-level system should be. In Sec. 2.2, most automated assembly systems generated sequential code or finite state machines describing the full assembly sequence. If one desires a system capable of opportunistically assembling parts available, more aspects of the high-level planning must be incorporated into the mid-level controller. To limit the scope of the discussion, let us consider the low-level controller taking joint-velocity setpoints, and the mid-level controller designed to handle one or more tasks.

One of the earliest approaches to translating tasks to robot motions was the *Resolved Motion Rate Control* (RMRC) [98] by Whitney (1969). RMRC was used to control a prostheses that had a desired trajectory for both the hand and the elbow. In continuous time, we can describe the deviation from the desired trajectory as

$$\boldsymbol{e}(t,\boldsymbol{q}) = \boldsymbol{p}(t) - \boldsymbol{f}(\boldsymbol{q}), \tag{3.1}$$

where $\boldsymbol{p} \in \mathbb{R}^{n_p}$ is a vector of the desired coordinates of the hand and elbow, $\boldsymbol{q} \in \mathbb{R}^{n_q}$ are the joint coordinates, and $\boldsymbol{f} : \mathbb{R}^{n_q} \to \mathbb{R}^{n_p}$ is a function describing the location of the relevant coordinates on the prostheses. Then we can find an expression for $\dot{\boldsymbol{q}}(t)$ by

$$\dot{\boldsymbol{q}}(t) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^{-1} \dot{\boldsymbol{p}}(t), \tag{3.2}$$

where the $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$ is often referred to as the task Jacobian. Starting with $\boldsymbol{e}$ sufficiently small, we can apply (3.2) to follow the desired coordinates, but we are not guaranteed to converge if disturbances occur.

The robot is said to be *redundant* with respect to the task if $n_p < n_q$. As this creates a non-square task Jacobian, its inverse is not defined. Whitney proposed that by defining an optimality criterion such as

$$\min \left(\dot{\boldsymbol{p}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right)^T \left(\dot{\boldsymbol{p}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right), \tag{3.3}$$

which the manipulator must satisfy during its motion, the inverse could be defined, e.g.:

$$\dot{\boldsymbol{q}}(t) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}^T\right)^{-1} \dot{\boldsymbol{p}}(t), \tag{3.4}$$

which holds for task Jacobians of full rank. This approach is also referred to as the generalized inverse [99] or pseudoinverse control [100] and can be velocity-resolved [101] or torque-resolved [102].

The *closed-loop inverse kinematics* (CLIK) formulation of RMRC addresses the lack of feedback control of the position error [103]–[105] and was, to the best of this author's knowledge, first described by Balestrino in 1984. In Fig. 3.3, a UR5 is tracking a trajectory. The robot did not start such that $\boldsymbol{e}(t, \boldsymbol{q}) = 0$, this is prime usecase for CLIK. CLIK works by designing the derivative of (3.1) to have an exponential convergence:

$$\dot{\boldsymbol{e}}(t, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = -\boldsymbol{K}\boldsymbol{e}(t, \boldsymbol{q}) \tag{3.5}$$

where $\boldsymbol{K}$ is positive definite. Expressed in terms of the joint velocity we have

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = -\boldsymbol{K}\boldsymbol{e}(t, \boldsymbol{q}) - \frac{\partial \boldsymbol{e}}{\partial t} \tag{3.6}$$

which can be solved using an optimization-based approach [93] or by using the pseudoinverse [101].

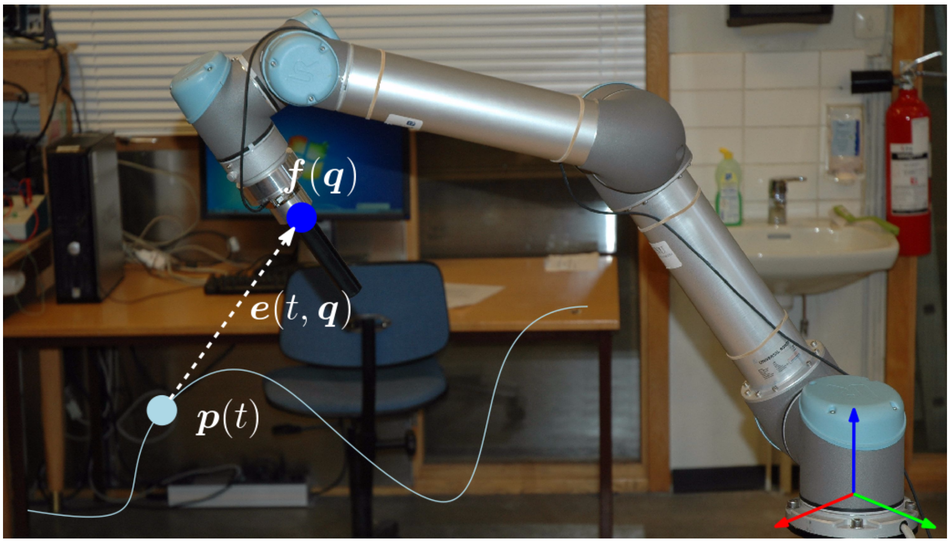

$\boldsymbol{f}(\boldsymbol{q})$
$\boldsymbol{e}(t, \boldsymbol{q})$
$\boldsymbol{p}(t)$

Figure 3.3: Visualization of the different parts of (3.1) for a UR5 tracking a trajectory.

A key concept of the RMRC is its generality and the possibility of sensor integration. The *task function* approach [106] by Samson et al. (1991) is a thorough presentation of sensor integration in a CLIK framework. A key insight of Samson et al. was that (3.1), which they refer to as the output function, can be described in terms of various variables, such as sensor input, and that the convergence criterion of CLIK imposes a control objective on the output function. They define a task as an output function with a control objective, for more information see Paper 7.

Due to the redundancy of manipulators with respect to their tasks, multiple tasks can be combined using either the null-space projection of higher-priority tasks [107], activation and deactivation of tasks [99], or by prioritization in an optimization problem [93]. One of the earliest cases of combining tasks by exploiting the redundancy of the manipulator with respect to the task was by Hanafusa et al. (1981) where a 7 DOF manipulator tracked a trajectory while avoiding an obstacle [108]. The trajectory tracking task was placed in the null-space of the collision avoidance task. The stability of the regulation problem when handling multiple tasks was described by Antonelli [109] in 2009, and the stability of the tracking problem is discussed in Paper 5. In 2011, Falco and Natale presented the stability of CLIK strategies for the regulation problem in the discrete time case in 2011. This was further described by Bjerkeng et al. [110]–[112] when accounting for tracking delay in the low-level controller.

Another notable early implementation of collision avoidance in robotics was the artificial potential field approach of Khatib [113]. Essentially one defines a potential field around obstacles that will repel the manipulator should it come near. A simplified formulation that gives a one-dimensional constraint is the local based approach by Faverjon [114] from 1987.

Forces can be accounted for in RMRC by indirect force control techniques. Where direct force control attempt to control the forces involved by controlling the motor torques, indirect force control techniques attempt to achieve a compliant behavior of the robot's interaction with its environment. This is similar to the behavior of the remote center of compliance, and can be used to make the robot more robust to positioning and part uncertainty and low assembly tolerances in assembly operations. Hogan introduced *impedance control* [90] in 1984 as a means of minimizing the interaction forces that may occur when the robot interacts with its environment. In impedance control, we attempt to control the positioning of the robot. *Admittance control* [115], introduced by Kazerooni et al. (1986), reverses the perspective and we adjust the positioning of the robot to accommodate an external force. Because of this, admittance control can be realized in systems where the low-level controller takes kinematic setpoints. When the static relationships are concerned, these are usually referred to as *stiffness* control and *compliance* control respectively [79]. As an example, consider that we have force sensors attached at the relevant coordinates of the previously presented prostheses, forming the vector $\boldsymbol{y}$. The prostheses is to follow the previously presented trajectory in a CLIK manner, and should exhibit compliance with respect to the forces. This means that

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = -\boldsymbol{K}\boldsymbol{e}(t, \boldsymbol{q}) - \frac{\partial \boldsymbol{e}}{\partial t} + \boldsymbol{y} \tag{3.7}$$

which gives the static relationship

$$\boldsymbol{K}\boldsymbol{e}(t, \boldsymbol{q}) = \boldsymbol{y} \tag{3.8}$$

if the desired trajectory is a point and the robot is static. This does not account for the dynamics of the transition from non-contact to contact, and more advanced techniques that describe the interaction at a task acceleration level may be necessary. The static relationship is a situation where we allow the robot to deviate from the desired position to accommodate the sensed force.

The previous paragraphs show how there have been momentous achievements in early robotics for task-based control. These results have then been further refined through the years, allowing thoroughly established theory for velocity-resolved continuous control of robots. In part, one can consider this as viewing the mid-level controller from the bottom-up. Seeing the underbelly of an idealized system that is above you. The other perspective, looking from the high-level controller, considers specification and formalization of what the mid-level controller is to achieve.

In 2007, De Schutter et al. described a *constraint-based task specification* [116]. If the transformation matrix from a reference frame $a$ to reference frame $b$ is $\boldsymbol{T}_a^b \in \mathbb{R}^{4\times 4}$, and one defines the transformation from world to end-effector as $\boldsymbol{T}_w^e$, transformation from the world to a geometric feature on a part described in terms of an auxiliary variable $\boldsymbol{x}$ called *feature* variable as $\boldsymbol{T}_w^f$, and one wishes to minimize an observed distance $y$ between the end-effector and the geometric feature, then De Schutter shows how the position-loop constraint

$$\boldsymbol{T}_w^e(\boldsymbol{q})\boldsymbol{T}_f^w(\boldsymbol{x})\boldsymbol{T}_e^f(y) = \boldsymbol{I}_{4\times 4} \tag{3.9}$$

can be used to formulate a joint-velocity control law. The constraint-based task specification is a general procedure for task specification and design that extends the task function approach. De Schutter et al. also incorporate virtual and artificial constraints based on the work of Mason [91]. A key insight of De Schutter et al. was formulating a consistent approach for handling geometric uncertainty, specifying how to handle sensor data with respect to the position-loop constraint, and the power of a specification method general enough to be applied to a variety of situations. This led to the development of *eTaSL/eTC* [93] by Aertbëlien et al. (2014), a constraint-based task specification language for continuous reactive control of the robot, as well as *rFSM* [117] by Klotzbücher and Bruyninckx (2012), a real-time safe finite state machine used for describing the discrete switching of robot tasks. As described in Paper 4, with eTaSL/eTC and rFSM, the high-level controller provides finite state machine descriptions of the intended behavior of the robot. As the robot accomplishes this behavior, or errors occur, the high-level controller would be notified and replanning would be performed.

Another task specification method was used in HighLAP. Namely the *manipulation primitives* first presented by Mosemann and Wahl in 2001 [47], and further formalized by Kröger et al. in 2010 [118]. The formalization describes how one designs task frames for continuous control of the robot, with well-defined switching behavior between control modes. *LightRocks* is a UML/P-based robot programming language by Thomas et al. [119] that has *elemental actions* reminiscent of the manipulation primitives. In this formalization, the mid-level controller processes the state chart, switching the elemental action to perform according to sensor events.

From an outsider's perspective, the two formalizations presented are closely related. The freedom presented by the work of De Schutter et al. allows for creating and combining complex tasks. The clarity and strictness presented by the work of Kröger et al. allows for a clearer picture of what the high-level controller should do. Other formalizations where the robot motions are described by a finite set of motion instructions exist, and are more closely related to the proprietary robot programming languages most robot manufacturers provide. Attempts have been made at standardizing these instructions, such as the *canonical robot command language* [120] by the National Institute of Science and Technology, USA (2016).

The previous paragraphs have focused on reactive control strategies. They base the choice of control input on the instantaneous situation that robot and its environment is in. The model predictive control approaches such as the model predictive path following controller presented in Paper 2 and Paper 3 that are based on the work of Faulwasser et al. [96], [97], [121], or the model predictive constraint-based task controller of Paper 7 attempt to exploit the future. The model predictive approaches are local planning systems that attempt to predict the motion of the robot to take future information into account. The work presented in this thesis only considers scenarios in which future sensor information cannot be predicted, and the controller is supplied with a path or constraint-based task specification from a higher-level controller.

Local planners are generally limited by the horizon they look ahead. This means that they may succumb to local minima such as not knowing how to move out of a a corner, or stop when an obstacle blocks its path (see Paper 2). This may be a desired behavior of the robot, or it may be indicative of the limits of local planning. Global planners can give solutions that would circumvent these issues. The combinatorial planning approaches give exact solutions for continuous space, but are known to be PSPACE-hard[1] even for simple 2D cases [122]. The sampling-based approaches either sample the joint space and define obstacles in joint space [123], or sample Cartesian space and provide a set of waypoints that another trajectory generation algorithm handles. The sampling-based approaches can be memory intensive due to the high dimensional spaces involved.

Despite these limitations, there have been attempts at *real-time adaptive motion planning* (RAMP) [124]. The idea is to apply one of the global planning approaches, that is guaranteed to work for structured environments, in an iterative fashion, thereby allowing the environment to vary. This has been considered with respect to robot dynamics [125], by deformation of an initial guessed path [126]–[128], by querying a pregenerated probabilistic roadmap [129], or a variety of other approaches. For these systems the intent of the high-level controller is usually considered as a desired list of configurations of the robot, or a desired list of relative configuration of parts. Examples of systems capable of real-time on-line trajectory planning include CHOMP [130] by Ratliff et al. (2009), STOMP [131] by Kalakhrisnan (2011), or the real-time trajectory generation using model predictive control [132] by Ardakani et al. (2015). Both CHOMP and STOMP have plugins for the popular ROS [28] motion planning package *MoveIt* [133].

[1]A PSPACE problem can be decided with no more than polynomial storage space (superset of NP).

## 3.4 Discussion and Future Work

The papers presented in this thesis contain contributions in: nonlinear speed observers for general mechanical systems, a mid-level model predictive path-following controller, a system architecture for CAD-based robotic assembly, implementation and timing analysis of an open-source ROS interface for KUKA robots, a stability proof of the tracking problem with task-priority inverse kinematic, and both reactive and model predictive closed-loop inverse kinematics controllers for constraint-based multiple task control.

The nonlinear speed observer of Paper 1, exhibits globally exponential convergence, a promising property as it is based on an observer applicable to a large variety of mechanical systems. Due to this, the goal was to see if a method could be created to automatically generate speed observers for any robot system, and it was expected that such a method would be beneficial in adaptive control of industrial manipulators. In practice, the observer was a very stiff dynamical system, and very small timesteps were necessary for simulation of a simple two DOF manipulator. Loria presented the article *Observers are unnecessary for Output-Feedback Control of Lagrangian Systems* [134], where the dirty-derivative filter was shown to be a sufficient alternative to speed observers. It was also observed that most commercially available industrial manipulators have kinematic control setpoints. For these reasons, the work was not pursued further.

The model predictive path following controller of Paper 2 and Paper 3 showed more promising results. Many industrial tasks are defined by a path rather than a trajectory. The focus is rather on the deviation of the end-effector from the path than its timing along the path. It also has interesting properties when faced with an obstacle. By tuning the length of the horizon and the weights in the model predictive controller, one can allow the end-effector to move around small obstacles, or stop when it arrives at a larger obstacle. When the industrial manipulator has kinematic setpoints, and one has access to a symbolic algorithmic differentiation tool such as CasADi [8], the model predictive controller can be used with any manipulator for which the parameters of the forward kinematics is known. This opened up an avenue into RMRC approaches, and its many related approaches. Specifically the constraint-based task-specification language eTaSL. The convergence of the model predictive path-following controller has been explored in the two DOF case [121], but the question of how to prove the convergence in the general case, both for redundant and non-redundant manipulators, is yet unanswered. Further work also includes implementing the model predictive path-following controller in CASCLIK.

The system architecture for constraint-based robotic assembly with CAD information of Paper 4 is an attempt at looking at the integration of a constraint-based task-specification language in a larger automated assembly system. The architecture includes novel features such as finite state machines based on the operational mode that the manipulator is in, and inference of parameters related to the assembly process. Integration of geometric information from CAD models of the assembly has been thoroughly researched, as presented in Sec. 2.2, but enabling complex sensor-based assembly tasks still remains a difficult prospect. By separating the parameters into two categories, *application parameters* related to the geometric information available in the CAD models, and *process parameters* that must be inferred from the models and the materials involved, the architecture attempts to simplify the process of generating controllers that can perform the assembly.

One of the key possibilities with constraint-based task specification is that multiple tasks can be combined and achieved simultaneously. Intuitively one can enforce joint limits, collision avoidance, and guarded motion in one and the same skill. When considering such situations, one is discussing the granularity of the architecture. What is the smallest elements involved? A collection of constraints, or a single task constraint? As the architecture was aimed at industrial assembly situations, the granularity was chosen at an atomic skill level. Each atomic skill was a single eTaSL script, defined by a collection of task constraints, that were known to work. The alternative, describing singular task constraints as the smallest element that the higher-level controller is to compose is not guaranteed to result in feasible skills, and it is difficult to evaluate beforehand whether the resulting skill can be achieved in the whole workspace of the robot with any possible sensor input. The chosen granularity makes it easier to create finite state machines of atomic skills into *composed skills*, where each state defines an operational mode of the controller. Operational mode in this context means that the state describes which atomic skill is active, or which form of continuous motion the robot is currently executing. Be it guarded Cartesian motion or compliant peg-in-hole insertion. By defining the composed skills as having each state as an operational mode of the robot, the number of states in the finite state machine is reduced, but the complexity is shifted to the transitions rather than the states. Classically in robotics, the finite state machine has states defined in terms of objectives to be achieved rather than the operational mode of the controller, e.g. “move to grasp”.

In some ways, the architecture can be considered to be limited by the freedom available. In this architecture, the mid-level controller is the composed skill executor, created with rFSM and eTaSL/eTC. The high-level controller is the process layer of the architecture. Constraint-based robot programming is closely related to the geometric constraints between features in

CAD. Intuitively this presents an opportunity to exploit the CAD information for generating the relevant skills, but a variety of problems arise when one wishes to go from idealized objects to real-life. For position-based control of parts, automatic generation of skills can be accomplished, such as the work of Somani et al. [135] (2016). They define assemblages in terms of CAD constraints, which allows for intuitive robot programming [136]. For tasks requiring sensor-based control, the problem of translating CAD information and constraints into tasks that can be accomplished with a constraint-based control remains elusive. The approach used in the system architecture is to consider eTaSL as a tool by which one can easily construct sensor-based assembly skills, still requiring a programmer for the skills. Instead of relying on CAD constraints, *assembly tasks* are defined in the application layer, for which a set of composed skills are feasible. The planning module is then only required to look through the set of composed skills associated with the assembly task when choosing the appropriate skill. Further work includes investigating automatic generation of composed skills from the CAD information.

CASCLIK was created as an attempt to investigate the flexibility of the system proposed by eTaSL/eTC. It follows the architecture of eTaSL, allowing a collection of tasks to be combined into a specification that can be executed by a variety of controllers. By using CasADi, algorithmic differentiation of arbitrary constraint expressions can be performed. This allows one to compare different underlying representations for the reference frames such as full transformation matrices with dual quaternions. CASCLIK is also meant for investigating whether a model predictive formulation of the controller is feasible and beneficial, and whether modification of the objective function can be of use. The original RMRC description by Whitney considers an objective function weighted with respect to an approximation of the kinetic energy of the system. Samson et al. furthers this discussion with respect to manipulability of the end-effector [106], which has also been addressed in a quadratic programming formulation without CLIK by Dufour et al. [137]. Model predictive controllers often define objectives to be achieved by the robot in terms of the cost function, see the cost function in Paper 2 or Paper 3. CASCLIK enables this by taking the cost expression as input to the instantiation of a controller. Further work also includes the implementation of indirect force control formulations such as an admittance controller [79] for the reactive control approaches, as well as torque-resolved controller formulations. The controller formulations should also be optimized for execution speed, as this is a prerequisite for using them in feedback control.

Researching control of robots is like searching for hay in a haystack. Robotics encompasses a variety of systems, and multitudinous articles from the early days of robotics to last Tuesday have presented relatable and useful results. From an outside perspective one may consider kinematic control of industrial manipulators to be solved, and the limitations simply being implementation. But then why are there not more complete automated assembly systems available?

Part of the reason appears to be the complexity of high-level planning as briefly discussed in the previous section. As one attempts to plan tasks in a higher-dimensional space for manipulators with an arbitrary number of joints, the complexity of the problem grows with the number of DOF of the manipulator. With dynamic unstructured environments, such as a busy work cell, real-time feedback planning may be required for which efficient solvers are still an issue. From the thought experiment in the beginning of the previous section, there is no unified approach to what should be transmitted to the mid-level controller. To a certain degree, one can say that the core requirements of robotics presented by Lozano-Perez [52] in 1983 are still not solved. When they are, if they ever will be, there is still the problems described by Hägele et al. of system integration and system design. Different robot manufacturers may provide different interfaces to the manipulator, and the low-level controller is often a black box that one has little insight into. As one nails down a rigid formalization of the low-level controller, the mid-level controller, or the high-level controller, one limits the possibilities of the other components. Integration of new sensor technology, new planning techniques, or new feedback control techniques into the same architecture is then more difficult.

Intuitively, the task function approach of Samson et al.[106] is a highly generalizable problem formulation. This makes it attractive as various systems and scenarios can be envisioned, not only in control of industrial manipulators but in control of robot platforms such as underwater swimming manipulators [138] or mobile robots [116], [139]. It can also be used with various interfaces to the low-level controller, at a kinematics, or dynamics level. And from the development of CASCLIK, it appears that only a small step is required from the reactive control formulation to a model predictive control formulation.

System integration may not be solved, but open-source middleware solutions such as OROCOS or ROS simplify the process of controlling robot systems and developing new software. Hardware standardizations such as the *Hardware Robot Information Model* [140] proposed by Acutronic Robotics (2018) that describe a standardization of ROS topics from sensors may further simplify sensor integration. Software such as Gazebo [141] provide interfacing between ROS and various physics engines. These tools allow a simplification of the system integration step that is unprecedented in robotics history, but also increase the variety of approaches to the problems as each research group still think and program in their own parlance.

When faced with the combinatorial explosion of system integration and possible methods of control, one cannot expect to create an ideal, or permanent solution to any robotics problem. Creating tools that are left to dust among textbooks and hard drives, or dead before they hit the market is a disquieting thought. Giving others insight and interfaces to our software enables others to learn from our accomplishments and failures, and perhaps usher in a new set of tools from which we can learn.

# Chapter 4

# Original Publications

# Paper 1 On the Globally Exponentially Convergent Immersion and Invariance Speed Observer for Mechanical Systems

M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, “On the globally exponentially convergent immersion and invariance speed observer for mechanical systems,” *American Control Conference (ACC)*, Seattle, WA, 2017, pp. 3294-3299.

# On the Globally Exponentially Convergent Immersion and Invariance Speed Observer for Mechanical Systems

Mathias Hauan Arbo*, Esten Ingar Grøtli† and Jan Tommy Gravdahl*
*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
† Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

***Abstract*—In this article we present a reformulation of the invariance and immersion speed observer of Astolfi et al. as applied to mechanical systems with bounded inertia matrices. This is done to explore the possibility of its practical implementation e.g. for 6 degrees-of-freedom industrial robots. The reformulation allows us find an explicit expression for one of the bounds used in the observer, and a constructive method for the second. We show that the observer requires either analytically or numerically solving at most $2n^2$ integrals, where $n$ is the number of generalized coordinates in the mechanical system.**

## I. Introduction

Industrial robots are typically equipped with encoders to measure the angles of the joints. These give accurate and steady measurements of the angles, but the speeds of these angles are not always available. Some systems have a tachometer, inertial sensors, or perform numerical differentiation of the joint position. These methods can introduce high-frequency noise or phase lag. An alternative method is to construct an observer for the nonlinear system.

Speed observers have been an important topic in robotics. One of the first references to a speed observer for mechanical systems is [1], where an asymptotic observer was used in a feedback situation with a trajectory tracking controller. In [2] a semi-globally exponentially stable observer based on passivity was presented. In [3] a globally exponentially convergent observer was established for general Euler-Lagrange systems. In [4] the observer was applied to systems with non-holonomic constraints and the system written in port-Hamiltonian form. In [5], a novel observer was presented with globally exponentially stable properties given that the inertia matrix has an upper bound. Speed observers are most notably featured in the topic of output-feedback control, where there are two main approaches: model-based approaches which utilize speed observers [5], [6], and filter-based approaches which use filters to replace speed observers [7]. In this article we are interested in the speed observer of [3]. The usage of the invariance and immersion observer for the output-feedback tracking scenario is outside the scope of this article, however we refer the reader to [8] where a globally exponentially stable trajectory controller that uses the immersion and invariance observer for a mechanical system in port-Hamiltonian form is presented.

In [9], Karagiannis et al. showed that a globally asymptotically convergent speed observer can be constructed for 2 degrees-of-freedom mechanical system if a certain partial differential equation admits analytical solutions. In [10] Karagiannis et al. presents a method of approximating such a partial differential equation using output filters and a dynamic scaling parameter. The deviation between the partial derivative of the approximated solution and the ideal solution is compensated for by the dynamic scaling parameter. In [3], Astolfi et al. applied this approximation method to general Euler-Lagrange systems to create a globally exponentially convergent speed observer. Astolfi et al. showed that there exist some bounds on the disturbances introduced by the deviation from the ideal partial differential equation. The speed observer was to be considered a proof of existence rather than a directly implementable method. The reason for this is that it requires the solution of a set of integrals that may not have closed-form solutions. To that end, they must be approximated numerically. Future developments in computational power may allow us to perform the necessary numerical integrations on-line. To that effect this article establishes how many integrals are needed at most. Given the Euler-Lagrange equations, the remaining difficulty is then how to define the necessary bounds on our deviation from the ideal partial differential equation.

In both [9] and [3], the speeds of the mechanical system were transformed by the Cholesky factorization of the inertia matrix, see the preliminary lemma of [3]. This transformation gives rise to a skew-symmetric property that simplifies the Lyapunov analysis. The basis of which stems from the skew-symmetry of the derivative of the inertia matrix. A similar transformation is performed on the port-Hamiltonian system in [4]. In this paper we show that, given the property that the inertia matrix has an upper bound, this transformation is not necessary. This excludes us from applying the observer to systems with an infinitely extending prismatic joint, but allows us to make an explicit bound on one of the disturbances. We give a constructive method of finding the other bound, a method that can also be applied to the observer of Astolfi et al., and we show how a naively implemented observer requires the evaluation of at most $2n^2$ integrals. This article is an attempt to explore the possibility of using the invariance and immersion globally exponentially convergent observer for general mechanical robots with bounded inertia matrices.

The paper is organized as follows: Section II contains two subsections, system and observer. The system subsection describes the system discussed in this paper and the properties assumed for it, and the observer subsection follows the proof of stability of our observer, a parallel to the proof of [3], and presents two lemmas for finding the bounds. Section III contains two sections, system description and results. Finally, Section IV contains the discussion and conclusion.

## II. Theory

### A. System

In this paper we consider an $n$ degrees-of-freedom robot described by the differential equation

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = u \tag{1}$$

where $q \in \mathbb{R}^n$ are the generalized coordinates, $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, $G(q) \in \mathbb{R}^n$ is a vector of potential forces, and $u \in \mathbb{R}^n$ is the control input. For any $q, x, y \in \mathbb{R}^n$ and $\lambda_i \in \mathbb{R}$ we assume the following properties hold

P1: $k_M I \geq M(q) \geq k_m I$
P2: $C(q,x)y = C(q,y)x$
P3: $C(q, \lambda_1 x_1 + \lambda_2 x_2)y = \lambda_1 C(q,x_1)y + \lambda_2 C(q,x_2)y$
P4: $\|C(q,x)\| \leq k_c \|x\|$
P5: $\dot{M}(q) = C(q,\dot{q}) + C(q,\dot{q})^T$.

with some known positive $k_M$, $k_m$, and $k_c$. The property P4 is not required for stability, but simplifies one of the bounds as will be shown in Lemma 1. P5 is to cancel the Coriolis matrices, for more information on applicability of P1 and P5 we refer the reader to [11]. The system admits the state-space representation:

$$\dot{y} = x \tag{2a}$$
$$\dot{x} = M(y)^{-1}(u - C(y,x)x - G(y)) \tag{2b}$$

with $y$ the measured coordinates, and $x$ being the unmeasured speeds of the generalized coordinates.

### B. Observer

For the system (2), we define the observer

$$\dot{\xi} = \alpha_1(\xi, s_y, s_x, y, u, r) \tag{3a}$$
$$\hat{x} = \xi + \beta(y, s_y, s_x) \tag{3b}$$
$$\dot{s}_y = \alpha_2(\hat{x}, s_y, s_x, y, r) \tag{3c}$$
$$\dot{s}_x = \alpha_3(\hat{x}, s_y, s_x, y, u, r) \tag{3d}$$
$$\dot{r} = \alpha_4(\hat{x}, s_y, s_x, y, r) \tag{3e}$$

where $\xi$, $\hat{x}$, $s_y$, $s_x$ and $r$ are the internal states of the observer. The output of the observer will be $s_x$, a filtered version of $\hat{x}$. We will define the dynamics $\alpha_i$ and mapping $\beta$ will so that the error

$$z = \hat{x} - x \tag{4}$$

has a globally exponentially stable origin. The proof of this follows the proof in [3], except that we use P5 instead of the preliminary lemma of [3], where $x$ in (2b) is transformed by the Cholesky factorization of the inertia matrix. For brevity we will omit the arguments of $\alpha_i$ when referred to in the following proof.

The dynamics of the error is found from (4) and taking the derivative of (3b) as

$$\begin{aligned}\dot{z} =& \alpha_1 + \frac{\partial\beta}{\partial y}x + \frac{\partial\beta}{\partial s_y}\alpha_2 + \frac{\partial\beta}{\partial s_x}\alpha_3 \\ &- M(y)^{-1}\left(u - C(y,x)x - G(y)\right)\end{aligned} \tag{5}$$

choosing $\alpha_1$ as

$$\begin{aligned}\alpha_1 =& M(y)^{-1}\left(u - C(y,\hat{x})\hat{x} - G(y)\right) \\ &- \frac{\partial\beta}{\partial y}\hat{x} - \frac{\partial\beta}{\partial s_y}\alpha_2 - \frac{\partial\beta}{\partial s_x}\alpha_3\end{aligned} \tag{6}$$

the estimation error dynamics becomes

$$\dot{z} = M(y)^{-1}\left(C(y,x)x - C(y,\hat{x})\hat{x}\right) - \frac{\partial\beta}{\partial y}z. \tag{7}$$

From P2 and adding and subtracting $C(y,x)\hat{x}$, we have

$$C(y,x)x - C(y,\hat{x})\hat{x} = -C(y,x)z - C(y,\hat{x})z \tag{8}$$

giving the dynamics

$$\dot{z} = -M(y)^{-1}\left(C(y,x) + C(y,\hat{x})\right)z - \frac{\partial\beta}{\partial y}z. \tag{9}$$

With a Lyapunov function $V(t,z) = \frac{1}{2}z^T M(y)z$, and using P5, we can see that solving $\frac{\partial\beta}{\partial y}$ such that

$$\frac{\partial\beta}{\partial y} = M(y)^{-1}\left(k_1 I - C(y,\hat{x})\right) \tag{10}$$

yields a globally exponentially convergent observer for $k_1 > 0$. But the partial differential equation is not always solvable, we follow the proof of [3] and approximate (10).

We define

$$H(y,\hat{x}) := M(y)^{-1}\left(k_1 I - C(y,\hat{x})\right) \tag{11}$$

as the ideal we want. Then following step 2 in [3], which uses a method from Karagiannis et al. in [10], we choose to model $\beta$ as the sum of $n$ integrals

$$\begin{aligned}\beta(y,s_y,s_x) :=& \int_0^{y_1(t)} H_1([y_1, s_{y2}, \ldots, s_{yn}], s_x)\mathrm{d}y_1 + \ldots \\ &+ \int_0^{y_n(t)} H_n([s_{y1}, \ldots, s_{yn-1}, y_n], s_x)\mathrm{d}y_n\end{aligned} \tag{12}$$

where the subscript $i$ of $H_i$ is the $i$th column of the matrix $H$, and $s_{yi}, y_i$ refers to the $i$th element of the vectors. Note that we are substituting $\hat{x}$ for the filtered state $s_x$, and $y$ for the filtered state $s_y$ for all but the integrated elements $y_i$. We will denote this as the column $H_i$ being a function of $y_i$, $s_{y1:n\setminus i}$, and $s_x$. This gives us the partial differential equation

$$\frac{\partial \beta}{\partial y} = \left[H_1([y_1, \dots, s_{yn}], s_x) \dots H_n([s_{y1}, \dots, y_n], s_x)\right]. \tag{13}$$

We can now also find the partial derivatives needed for $\alpha_1$, starting with

$$\frac{\partial \beta}{\partial s_{yi}} = \sum_{j=0, j\neq i}^{n} \int_0^{y_j(t)} \frac{\partial H_j}{\partial s_{yi}}(y_j, s_{y1:n\setminus j}, s_x) \mathrm{d}y_j \tag{14}$$

which requires at most we $n(n-1)$ integrals. Secondly we have

$$\frac{\partial \beta}{\partial s_{xi}} = \sum_{j=0}^{n} \int_0^{y_j(t)} \frac{\partial H_j}{\partial s_{xi}}(y_j, s_{y1:n\setminus j}, s_x) \mathrm{d}y_j \tag{15}$$

similarly to (14), it requires evaluating $n^2$ integrals.

We define the deviation of our modeled $\beta$ from the ideal as

$$\begin{aligned} &H_i(y, \hat{x}) - H_i(y_i, s_{y1:n\setminus i}, s_x) = \\ &\quad H_i(y, \hat{x}) - H_i(y, s_x) + H_i(y, s_x) - H_i(y_i, s_{y1:n\setminus i}, s_x) \\ &\quad = \Delta_{x,i}(y, s_x, e_x) + \Delta_{y,i}(y, s_x, e_y) \end{aligned} \tag{16}$$

where $i$ refers to the column vector of the matrices $\Delta_x$ and $\Delta_y$, and we have defined the errors

$$e_y := s_y - y \tag{17}$$

$$e_x := s_x - \hat{x} \tag{18}$$

which, from (13) and (16), gives us

$$\frac{\partial \beta}{\partial y} = H(y, \hat{x}) - \Delta_x(y, s_x, e_x) - \Delta_y(y, s_y, e_y). \tag{19}$$

As (13) is the same as (11) when $y = s_y$ and $\hat{x} = s_x$, we see that when $e_y = 0$ and $e_x = 0$ then $\Delta_x = 0$ and $\Delta_y = 0$. In combination with the fact that the mappings are smooth this ensures that there exists some mappings $\bar{\Delta}_x$, $\bar{\Delta}_y$ such that

$$\Delta_x(y, s_x, e_x) = \bar{\Delta}_x(y, s_x, e_x)e_x \tag{20a}$$

$$\Delta_y(y, s_y, e_y) = \bar{\Delta}_y(y, s_y, e_y)e_y. \tag{20b}$$

Theoretically we can see that $\bar{\Delta}_x$ and $\bar{\Delta}_y$ can be found by taking the Taylor series around $e_x = 0$ and $e_y = 0$ and factoring out $e_x$ or $e_y$. But as many systems contain trigonometric expressions, this is potentially an infinite series that is not easy to implement. The following lemma gives a constructive method of finding $\|\Delta_x\|$ related to $\|e_x\|$.

**Lemma 1.** *Given a matrix $\Delta_x(y, s_x, e_x)$ as defined in (16), and the properties* P1, P3, *and* P4, *then*

$$\|\Delta_x(y, s_x, e_x)\| \leq \frac{k_c}{k_m}\|e_x\| \tag{21}$$

*Proof.* From (16) we have

$$\begin{aligned} \Delta_x(y, s_x, e_x) &= H(y, s_x - e_x) - H(y, s_x) \\ &= M(y)^{-1}\left(-C(y, s_x - e_x) + C(y, s_x)\right) \\ &= M(y)^{-1}\left(C(y, e_x)\right) \end{aligned} \tag{22}$$

where we have used P3 to arrive at the last line. And thus

$$\|\Delta_x(y, s_x, e_x)\| \leq \left\|M(y)^{-1}\right\| \|C(y, e_x)\| \tag{23}$$

which, given P1 and P4 becomes (21). □

When P4 is available, Lemma 1 gives an explicit bound on $\Delta_x$. For $\Delta_y$ the partial substitution of $s_y$ for $y$ for all but element $y_i$ complicates matters. The following lemma is more general and is used to find bounds that relate $\|\Delta_x\|$ or $\|\Delta_y\|$ to $\|e_x\|$ or $\|e_y\|$.

**Lemma 2.** *Given a function $\Delta : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$ that is continuously differentiable, where*

$$\Delta(x, y, e) = 0 \tag{24}$$

*if*

$$e = 0 \tag{25}$$

*and*

$$\left\| \sup_{e\in\mathbb{R}^n} \frac{\partial \Delta_{ij}(x, y, e)}{\partial e} \right\|_2 = \bar{\Delta}_{ij}(x, y, e) \tag{26}$$

*where the subscript $ij$ refers to an element in the matrix. Then*

$$\|\Delta(x, y, e)\|_2 \leq \left\|\bar{\Delta}(x, y, e)\right\|_F \|e\|_2 \tag{27}$$

*Proof.* From the definition of the matrix 2 norm and Frobenius norm we have

$$\|\Delta(x, y, e)\|_2 \leq \|\Delta(x, y, e)\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} \Delta_{ij}(x, y, e)^2} \tag{28}$$

from (26), there exists a supremum such that

$$\begin{aligned} |\Delta_{ij}(x, y, e)| &\leq \left| \left( \sup_{e\in\mathbb{R}^n} \frac{\partial \Delta_{ij}(x, y, e)}{\partial e} \right) e \right| \\ &\leq \bar{\Delta}_{ij}(x, y, e)\, \|e\|_2 \end{aligned} \tag{29}$$

and putting this into the (28) gives

$$\|\Delta(x, y, e)\|_2 \leq \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} \bar{\Delta}_{ij}(x, y, e)^2}\, \|e\|_2 \tag{30}$$

which is equal to (27). □

Continuing on the stability proof, we substitute (19) in (9)

$$\begin{aligned} \dot{z} = &-M(y)^{-1}\left(C(y, x) + C(y, \hat{x})\right) z \\ &- H(y, \hat{x})z + (\Delta_y + \Delta_x)z \end{aligned} \tag{31}$$

and using (11) we get

$$\dot{z} = -M(y)^{-1}\left(k_1 I + C(y, x)\right) z + (\Delta_y + \Delta_y)z \tag{32}$$

The matrices $\Delta_x$ and $\Delta_y$ act as disturbances on $z$ that we will dominate with a dynamic scaling $r$. We define a new scaled variable as

$$\eta = \frac{z}{r} \tag{33}$$

with the derivative

$$\dot{\eta} = \frac{\dot{z}}{r} - \frac{\dot{r}}{r}\eta. \tag{34}$$

We define the Lyapunov function $V_1(t,\eta) = \frac{1}{2}\eta^T M(y)\eta$ so as to cancel the Coriolis matrices. Using (32), (34), and P5, we get

$$\dot{V}_1 = -k_1 \|\eta\|^2 + \eta^T M(y)(\Delta_y + \Delta_x)\eta - \frac{\dot{r}}{r}\eta^T M(y)\eta \quad (35)$$

using P1

$$\begin{aligned}\dot{V}_1 &\le -k_1 \|\eta\|^2 + \|M(y)(\Delta_y + \Delta_x)\| \|\eta\|^2 - \frac{\dot{r}}{r}k_m \|\eta\|^2 \\ &\le -\left(\frac{k_1}{2} + k_m\frac{\dot{r}}{r} - \frac{1}{2k_1}\|M(y)(\Delta_y+\Delta_x)\|^2\right)\|\eta\|^2 \\ &\le -\left(\frac{k_1}{2} + k_m\frac{\dot{r}}{r} - \frac{1}{k_1}(\|M(y)\Delta_y\|^2 + \|M(y)\Delta_x\|^2\right)\|\eta\|^2\end{aligned} \quad (36)$$

where the second inequality is found using Young's inequality with factor $k_1$. Choosing the dynamics of $r$ as

$$\dot{r} = -\frac{k_1}{4k_m}(r - c_r) + \frac{r}{k_1 k_m}\left(\|M(y)\Delta_y\|^2 + \|M(y)\Delta_x\|^2\right) \quad (37)$$

with $r(t_0) > c_r > 0$ and $r(t) > c_r > 0$ gives

$$\dot{V}_1 \le -\left(\frac{k_1}{2} - \frac{k_1}{4}\frac{r-c_r}{r}\right)\|\eta\|^2 \le -\frac{k_1}{4}\|\eta\|^2. \quad (38)$$

This gives global exponential stability of $\eta(t)$, and in turn we need boundedness of $r(t)$ to ensure global exponential stability of $z(t)$. The choice of placing a parameter $c_r$ as a lower bound on $r$ is inspired by [5].

We are going to create the Lyapunov functions

$$V_2(t,\eta,e_y,e_x) = V_1(t,\eta) + \frac{1}{2}(e_y^T e_y + e_x^T e_x) \quad (39)$$

$$V_3(t,\eta,e_y,e_x,r) = V_2(t,\eta,e_y,e_x) + \frac{1}{2}(r-c_r)^2. \quad (40)$$

From the definition of the errors (17) and (18), and taking the derivative of (3b), we have

$$\dot{e}_y = \alpha_2 - x \quad (41)$$

$$\dot{e}_x = \alpha_3 - \alpha_1 + \frac{\partial\beta}{\partial y}x + \frac{\partial\beta}{\partial s_y}\alpha_2 + \frac{\partial\beta}{\partial s_x}\alpha_3 \quad (42)$$

choosing

$$\alpha_2 = \hat{x} - \psi_1 e_y \quad (43)$$

$$\alpha_3 = M(y)^{-1}(u - C(y,\hat{x})\hat{x} - G(y)) - \psi_2 e_x \quad (44)$$

where $\psi_1$ and $\psi_2$ are scalar gain functions, with our chosen $\alpha_1$, (6), we have

$$\dot{e}_y = z - \psi_1 e_y \quad (45)$$

$$\dot{e}_x = \frac{\partial\beta}{\partial y}z - \psi_2 e_x. \quad (46)$$

The time derivative of $V_2$ is therefore

$$\dot{V}_2 = \dot{V}_1 + r e_y^T\eta + r e_x^T\frac{\partial\beta}{\partial y}\eta - \psi_1 e_y^T e_y - \psi_2 e_x^T e_x \quad (47)$$

$$\le -\left(\frac{k_1}{4} - 1\right)\|\eta\|^2 - \left(\psi_1 - \frac{r^2}{2}\right)\|e_y\|^2 - \left(\psi_2 - \frac{r^2}{2}\left\|\frac{\partial\beta}{\partial y}\right\|^2\right)\|e_x\|^2 \quad (48)$$

where we have used Young's inequality. From (37), we get the time derivative of $V_3$ as

$$\dot{V}_3 = \dot{V}_2 - \frac{k_1}{4k_m}(r-c_r)^2 + \frac{r(r-c_r)}{k_1 k_m}\left(\|M(y)\Delta_y\|^2 + \|M(y)\Delta_x\|^2\right). \quad (49)$$

As we have established (20a) and (20b), or similarly appropriate bounds through Lemma 1 and Lemma 2, we arrive at the inequality

$$\dot{V}_3 \le \dot{V}_2 - \frac{k_1}{4k_m}(r-c_r)^2 + \frac{r^2 k_M^2}{k_1 k_m}\left(\|\bar{\Delta}_y\|^2\|e_y\|^2 + \|\bar{\Delta}_x\|^2\|e_x\|^2\right) \quad (50)$$

where we have used the fact that $r^2 > r(r-c_r)$ for $r > c_r > 0$. Collecting the terms from (48) we get

$$\begin{aligned}\dot{V}_3 \le &-\left(\frac{k_1}{4} - 1\right)\|\eta\|^2 - \left(\psi_1 - \frac{r^2}{2} - \frac{r^2 k_M^2}{k_1 k_m}\|\bar{\Delta}_y\|^2\right)\|e_y\|^2 \\ &-\left(\psi_2 - \frac{r^2}{2}\left\|\frac{\partial\beta}{\partial y}\right\|^2 - \frac{r^2 k_M^2}{k_1 k_m}\|\bar{\Delta}_x\|^2\right)\|e_x\|^2 \\ &-\frac{k_1}{4k_m}(r-c_r)^2\end{aligned} \quad (51)$$

we can see that if we choose

$$\psi_1 = k_2 + \frac{r^2}{2} + \frac{r^2 k_M^2}{k_1 k_m}\|\bar{\Delta}_y\|^2 \quad (52)$$

$$\psi_2 = k_3 + \frac{r^2}{2}\left\|\frac{\partial\beta}{\partial y}\right\|^2 + \frac{r^2 k_M^2}{k_1 k_m}\|\bar{\Delta}_x\|^2 \quad (53)$$

we get

$$\dot{V}_3 \le -\left(\frac{k_1}{4} - 1\right)\|\eta\|^2 - k_2\|e_y\|^2 - k_3\|e_x\|^2 - \frac{k_1}{4k_m}(r-c_r)^2 \quad (54)$$

where we choose $k_1 > 4$. This gives $\dot{V}_3 \le 0$, which ensures that $r(t)$ is bounded. This in turn gives us global exponential convergence of $z(t)$.

As shown in [9], the difference between the solution to the ideal partial differential equation (10) and our approximation (12) is such that the states from the output filters, $s_y$ and $s_x$ give the best estimates of the system states from the "perspective" of our observer. One way to think of this is to see that $\xi$ would have been the states of our ideal observer, with output defined by $\hat{x}$. We cannot solve the differential equation required to render $\beta$ of the output function as we want it. So we give it dynamics $s_x$ and $s_y$ that filter the states to compensate for the disturbances introduced by the approximation. This means that $s_y$, and $s_x$ are the output of our observer.

## III. Simulation

### A. System description

We consider the 2 degrees-of-freedom system used in [3], with $y_1$ and $y_2$ defined as illustrated in Fig.1. We define

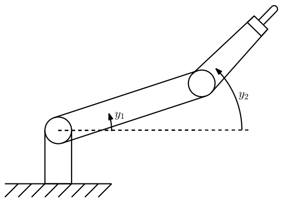


Fig. 1. Two-link manipulator with revolute joints.

TABLE I
INITIAL STATES OF OBSERVER AND ROBOT

| $y(0)$ | $x(0)$ | $\xi(0)$ | $s_y(0)$ | $s_x(0)$ | $r(0)$ |
|---|---|---|---|---|---|
| $[0,0]^T$ | $[-0.29, 6.66]^T$ | $[-8.45, 68.33]^T$ | $[1,2]^T$ | $[10,20]^T$ | 0.1 |

TABLE II
ROBOT AND OBSERVER PARAMETERS

| $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $k_1$ | $k_2$ | $k_3$ | $c_r$ |
|---|---|---|---|---|---|---|---|---|
| 0.9698 | 0.1575 | 0.264 | 1.74 | 0.44 | 40 | 8 | 2.5 | 0.001 |

$c(x,y) = \cos(x-y)$, $s(x,y) = \sin(x-y)$, $D(x,y) = c_1c_2 - c_3^2 c(x,y)^2$, and the system is described by

$$M(y) = \begin{bmatrix} c_1 & c_3 c(y_1,y_2) \\ c_3 c(y_1,y_2) & c_2 \end{bmatrix} \tag{55a}$$

$$C(y,x) = \begin{bmatrix} 0 & -c_3 s(y_1,y_2)x_1 \\ c_3 s(y_1,y_2)x_2 & 0 \end{bmatrix} \tag{55b}$$

$$G(y) = g\begin{bmatrix} c_4\cos(y_1) \\ c_5\cos(y_2) \end{bmatrix} \tag{55c}$$

with

$$c_1 = I_1 + I_2 + m_1 L_{c_1}^2 + m_2(L_1^2 + L_{c_2}^2) \tag{56a}$$

$$c_2 = 2m_2 L_{c_2} L_1,\ c_3 = m_2 L_{c_2}^2 + I_2 \tag{56b}$$

$$c_4 = g(L_{c_1}(m_1 + M_1) + L_1(m_2 + M_2))g \tag{56c}$$

$$c_5 = gL_{c_2}(m_2 + M_2) \tag{56d}$$

where the links are of length $L_i$, with link masses $M_i$ at $L_{c_i}$, and masses $m_i$ at the joints. We have

$$k_m = \lambda_{\min}(M_0)),\quad k_M = \lambda_{\max}(M_0),\quad k_c = c_3 \tag{57}$$

where $k_m$ and $k_M$ stem from the eigenvalues $\lambda$ of $M_0$, the inertia matrix $M$ at the origin $[y_1, y_2] = [0,0]$, and using the method of [12] we get $k_C$.

Using (11) we have:

$$H(y,x) = \begin{bmatrix} \frac{c_2k_1 + \frac{c_3^2 x_2}{2} s(2y_1,2y_2)}{D(y_1,y_2)} & \frac{c_3(c_2x_1 s(y_1,y_2) - k_1 c(y_1,y_2))}{D(y_1,y_2)} \\ -\frac{c_3(c_1x_2 s(y_1,y_2) + k_1 c(y_1,y_2))}{D(y_1,y_2)} & \frac{c_1k_1 - \frac{c_3^2 x_1}{2} s(2y_1,2y_2)}{D(y_1,y_2)} \end{bmatrix} \tag{58}$$

from which $\Delta_x$, and $\Delta_y$ can be constructed using (16). A simple albeit overestimating method of finding $\bar{\Delta}_{ij}$ from a symbolically calculated $\frac{\partial \Delta_{ij}}{\partial e}$ is the sum of the supremum of the terms of the expression. E.g. if $\frac{\partial \Delta_{ij}}{\partial e} = f_1 - f_2 + f_3$ then we choose $\bar{\Delta}_{ij} = \|f_1\|_\infty + \|f_2\|_\infty + \|f_3\|_\infty$. The initial states of the robot and the observer are given in Table I, and the robot parameters are given in Table II. The initial states of the robot and the initial estimate $s_x$ were chosen so as to coincide with the initial states in [3].

## B. Results

The graphs show the observer of this paper with parameters specified in Table II as well as that of [3] with $k_1 = 10$. In Fig.2 we see that the filtered angles $s_y$ converge to the actual trajectories $y$ over time. In Fig.3 we can see that the filtered speeds $s_x$ converge to the actual speeds $x$ over time. Note that with the same gains as in [3] our observer takes a longer time to converge. It appears that we require larger gains to achieve the same effect as in [3]. In Fig.5 we attempt to show our reasoning for choosing a small initial $r(t_0)$, and using $c_r$ to allow an $r$ lower than 1. The graph shows the norm of $[\dot{\xi}, \dot{s}_y, \dot{s}_x, \dot{r}]$ evaluated for varying $r$ at the initial state. With larger $r$ the observer differential equation is stiffer, requiring a small timestep to remain accurate. This trend was observed for many states other than the initial state as well. To simplify simulation, we chose a sufficiently small $r(t_0)$ so as to remain in the region where $r < 10^{-1}$.

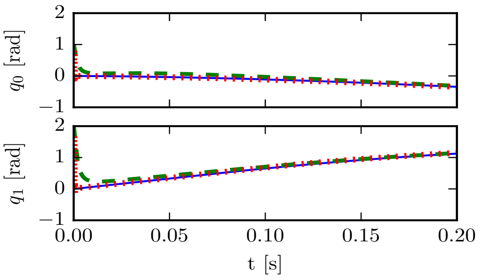


Fig. 2. Joint angles over time. The blue line is the actual angles, the dashed green line is $s_y$, and the dotted red line is the filtered angles from the observer in [3]. Note that this is shown for the timeframe 0 to 0.2 s.

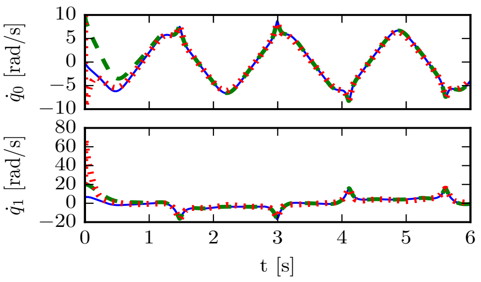


Fig. 3. Joint speeds over time. The blue line is the actual speeds, the dashed green line is $s_x$, and the dotted red line is the estimated speeds of the observer in [3].

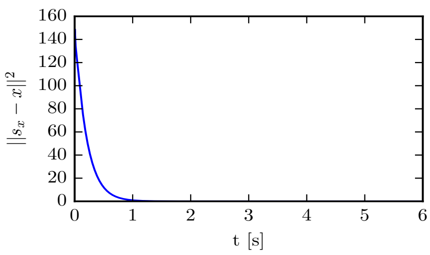


Fig. 4. $\|s_x - x\|^2$ over time.

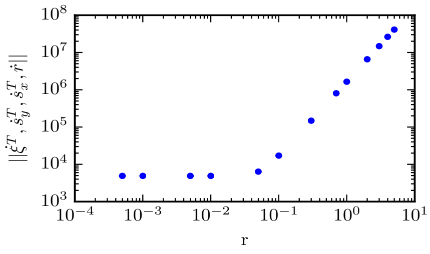


Fig. 5. The norm of $[\dot{\xi}^T, \dot{s}_y^T, \dot{s}_x^T, \dot{r}]^T$ with respect to $r$. Evaluated at the initial state of the observer described in Table I.

## IV. Discussion and Conclusion

### A. Discussion

Our alternative formulation of the observer does not come for free, we use a Lyapunov function $V_1$ that depends on the inertia matrix. This means that unlike [3] we require an upper bound on the inertia matrix, a property that does not hold for mechanical systems with inifinitely extending prismatic joints. Further work into relating the observer presented in this article to that of [3] may give explicit bounds for the disturbances. Lemma 2 can also be used for the disturbance bounds in [3].

As the second Lemma relies on finding the supremums of nonlinear equations, it is not as practical as Lemma 1. Investigating the general structure of $\Delta_y$ with respect to some classes of robotic systems, e.g. consisting of revolute joints defined by the Denavit-Hartenberg convention, might give rise to explicit bounds.

We can see that if all the integrals of (12) are performed numerically, which would allow us to generate an observer from only the symbolic system equations, naively implemented, we will end up with $2n^2$ numerical integrals from our approximation of $\beta$, $\frac{\partial \beta}{\partial s_y}$, and $\frac{\partial \beta}{\partial s_x}$ evaluated at each timestep. This might be cumbersome even with optimized methods of performing the integrals and parallelising the effort. Further work evaluating what to do when parts of the differential equation is solvable may reduce the number of numeric integrals needed. The exact number of integrals required depends on how one defines the generalized coordinates of the robot, and the mechanical structure of the system. For example, if the second joint in our example was defined relative to the first joint angle instead of the horizontal line, $H(y, \hat{x})$ would not depend on $y_1$ and the corresponding integrals would be zero.

### B. Conclusion

In this article we presented the observer of Astolfi et al. from [3] reformulated so as to give more intuitive internal states. These make it easier to provide an explicit method in Lemma 1 and constructive method in Lemma 2 for defining the necessary bounds. The method of Lemma 2 is sufficiently general to be applied to the observer of [3]. A result of our reformulation of the observer is that it requires an upper bound on the inertia matrix, a property that Astolfi et al. did not require. Through our approximation of a partial differential equation, the observer requires evaluation of $2n^2$ integrals.

## V. Acknowledgements

The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

## References

[1] S. Nicosia and P. Tomei, “Robot Control by Using Only Joint Position Measurements,” *IEEE Transactions on Automatic Control*, vol. 35, no. 9, pp. 1058–1061, 1990.

[2] H. Berghuis and H. Nijmeijer, “A Passivity Approach to Controller-Observer Design for Robots,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 6, pp. 740–754, 1993.

[3] A. Astolfi, R. Ortega, and A. Venkatraman, “A globally exponentially convergent immersion and invariance speed observer for n degrees of freedom mechanical systems,” in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, dec 2009, pp. 6508–6513.

[4] ——, “A globally exponentially convergent immersion and invariance speed observer for mechanical systems with non-holonomic constraints,” *Automatica*, vol. 46, no. 1, pp. 182–189, 2010.

[5] Ø. N. Stamnes, O. M. Aamo, and G.-O. Kaasa, “Global Output Feedback Tracking Control of Euler-Lagrange Systems,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 215–220, 2011.

[6] B. Xian, M. S. De Queiroz, D. M. Dawson, and M. L. McIntyre, “A discontinuous output feedback controller and velocity observer for nonlinear mechanical systems,” *Automatica*, vol. 40, no. 4, pp. 695–700, 2004.

[7] A. Loria, “Observers are Unnecessary for Output-Feedback Control of Lagrangian Systems,” *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 905–920, 2016.

[8] J. G. Romero, R. Ortega, and I. Sarras, “A Globally Exponentially Stable Tracking Controller for Mechanical Systems Using Position Feedback,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 818–823, mar 2015.

[9] D. Karagiannis, D. Carnevale, A. Astolfi, and S. Member, “Invariant Manifold Based Reduced-Order Observer Design for Nonlinear Systems,” *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2602–2614, 2008.

[10] D. Karagiannis and A. Astolfi, “Observer Design for a Class of Nonlinear Systems Using Dynamic Scaling with Application to Adaptive Control,” *47th IEEE Conference on Decision and Control*, pp. 2314–2319, 2008.

[11] P. J. From, I. Schjølberg, J. T. Gravdahl, K. Y. Pettersen, and T. I. Fossen, “On the Boundedness Property of the Inertia Matrix and Skew-Symmetric Property of the Coriolis Matrix for Vehicle-Manipulator Systems,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 134, no. 4, 2012.

[12] R. Kelly, V. Santibañez, and A. Loría, *Control of Robot Manipulators in Joint Space*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer-Verlag, 2005.

# Paper 2 On Model Predictive Path Following and Trajectory Tracking for Industrial Robots

M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, “On model predictive path following and trajectory tracking for industrial robots,” *13th IEEE Conference on Automation Science and Engineering (CASE)*, Xi’an, 2017, pp. 100-105.

# On Model Predictive Path Following and Trajectory Tracking for Industrial Robots

Mathias Hauan Arbo*and, Esten Ingar Grøtli† and Jan Tommy Gravdahl*
*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
†Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

***Abstract*—In this article the model predictive path following controller and the model predictive trajectory tracking controller are compared for a robotic manipulator. We consider both the Runge-Kutta and collocation based discretization. We show how path-following can stop at obstructions in a way trajectory tracking cannot. We give simulations for a two-link manipulator, and discuss the real-time viability of our implementations.**

## I. Introduction

Modern industrial robots weld, grind, screw, measure, film, paint, pick and place, and perform other tasks that require the robot to follow arbitrary paths in space. For a typical robot cell, we simplify our work as robotics engineers by having enclosed, structured workspaces with no obstructions. For dedicated large-scale production of a small set of products, this is easily achieved. Small-scale manufacturing has a large variety of products and each product series is produced in low-scale or on-demand. Flexible robot cells are moved around in cluttered workspaces and require rapid prototyping of the paths. This means the control strategies must handle obstructions without sacrificing quality of the product.

We consider model predictive control (MPC), as it handles constraints on states and rudimentary obstructions. This article considers known obstructions, and focuses on the difference between path-following and trajectory tracking.

In trajectory tracking model predictive control (TT-MPC), the robot is to follow a path with an explicit path-timing. The trajectory may even incorporate constraints on the torques or velocities of the robot through optimal control problem (OCP) approaches such as [1], where a time-optimal path timing law under constraints is generated. The path timing law specifies the relation between the desired path and time, while accounting for state constraints during the execution of the path. The OCP is solved under the assumption that the robot is moving along the path, and is an open-loop approach to the constraint handling. In [2] this was extended to also allow constraints on the acceleration and inertial forces at the end-effector.

In [3], a model predictive path-following controller (MPFC) that handles both path-timing and error from path in the same OCP is described. In [4], the MPFC is shown to converge to the path given terminal constraints without needing terminal penalties. In [5] the MPFC is implemented on a KUKA LWR IV robot, without end penalty or a terminal constraint. This is done with the ACADO framework [6], which uses a sequential programming method (SQP), iteratively solving quadratic programs approximating the nonlinear program using the qpOASES active set solver.

In [7], a real-time MPFC scheme for contouring control of an x-y table is described. Here a linear time varying approximation of the dynamics is used to define a QP which is solved using an active-set solver. In [8], this is implemented on an x-y table, and the MPFC outperformed both a similarly implemented TT-MPC, and the industry standard cascaded PI controlled set-point controller operating at a higher sampling frequency.

In [9], an MPFC is applied to a tower crane. The OCP is solved using the gradient projection method, an indirect method where Pontryagin's Maximum Principle is solved for the OCP without inequality constraints on the states. Instead slack variables are introduced to implicitly handle the inequality constraints.

In this article we

- draw attention to differences in MPFC and TT-MPC behavior, with and without obstructions,
- compare the Runge-Kutta and collocation integration method for the two strategies,
- solve the nonlinear programs (NLP) for the control strategies using the interior point solver IPOPT,
- and discuss the framework for real-time applications.

## II. Theory

### A. System

The robot is an $n$ degrees-of-freedom system with the state-space representation

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{x}(t) \tag{1a}$$

$$\dot{x}(t) = \boldsymbol{f_x}(\boldsymbol{y}(t), \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1b}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ are the generalized coordinates, $\boldsymbol{x} \in \mathbb{R}^n$ are generalized velocities, and $\boldsymbol{u} \in \mathbb{R}$ are the inputs. The function $\boldsymbol{f_x}$ describes acceleration. We assume the robot has known forward kinematics, allowing us to define a point of interest $\boldsymbol{p}$ on the robot such that

$$\boldsymbol{p}(t) = \boldsymbol{f_p}(y(t)) \tag{2}$$

where $\boldsymbol{f_p}(\cdot) \in \mathbb{R}^{n_p}$ is found from the forward kinematics. We describe the $\mathcal{C}^1$ reference path as $\boldsymbol{\varrho}(\cdot) \in \mathbb{R}^{n_p}$, defined in the same frame as $\boldsymbol{p}$.

The path-timing variable $s$ moves from 0 to $s_f$. The TT-MPC assumes $s = t$ for $t < s_f$ and $s = s_f$ otherwise. The MPFC controls $s$ through the path-timing dynamics, which we model as a double integrator

$$\begin{bmatrix} \dot{s}(t) \\ \ddot{s}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} s(t) \\ \dot{s}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v(t) := \boldsymbol{f_s}(s, \dot{s}, v), \tag{3}$$

with piecewise constant input $v(\cdot) \in \mathbb{R}$. To ensure that we never move backwards along the path, and that we have a maximum along path speed, we constrain $\dot{s} \in [0, \dot{s}_u]$. For more information on choice of the path-timing dynamics, we refer the reader to [4] and [9].

For the MPFC we define the extended state $\boldsymbol{\xi} = [\boldsymbol{y}^T, \boldsymbol{x}^T, s, \dot{s}]^T$, input $\boldsymbol{w} = [\boldsymbol{u}^T, v]^T$, and dynamics

$$\dot{\boldsymbol{\xi}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{f_x}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)) \\ \boldsymbol{f_s}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)) \end{bmatrix} = \boldsymbol{f_\xi}(\boldsymbol{\xi}(t), \boldsymbol{w}(t)). \tag{4}$$

Similarly, for the TT-MPC we define the extended state $\boldsymbol{\chi} = [\boldsymbol{y}^T, \boldsymbol{x}^T]$ with dynamics

$$\dot{\boldsymbol{\chi}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{f_x}(\boldsymbol{\chi}(t), \boldsymbol{u}(t)) \end{bmatrix} = \boldsymbol{f_\chi}(\boldsymbol{\chi}(t), \boldsymbol{u}(t)). \tag{5}$$

We define the deviation from the path as

$$\boldsymbol{e}_{pf}(t) := \boldsymbol{f_p}(\boldsymbol{y}(t)) - \boldsymbol{\varrho}(s(t)). \tag{6}$$

for the MPFC, and

$$\boldsymbol{e}_{tt}(t) := \boldsymbol{f_p}(\boldsymbol{y}(t)) - \boldsymbol{\varrho}(t) \tag{7}$$

for the TT-MPC. For $s$ to converge to $s_f$ we also define the path-timing error

$$e_s(t) := s(t) - s_f. \tag{8}$$

## B. Optimal Control Problem

With the previously defined dynamics and errors, we can describe the OCP for the MPFC as

$$\min_{\boldsymbol{e}_{pf}, \dot{\boldsymbol{e}}_{pf}, e_s, \boldsymbol{u}, v} \int_{t_k}^{t_k+T} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) \mathrm{d}\tau \tag{9a}$$

s.t.:

$$\dot{\bar{\boldsymbol{\xi}}}(\tau) = \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) \tag{9b}$$

$$\bar{\boldsymbol{\xi}}(t_k) = \boldsymbol{\xi}(t_k) \tag{9c}$$

$$\bar{\boldsymbol{\xi}}(\tau) \in [\bar{\boldsymbol{\xi}}_l, \bar{\boldsymbol{\xi}}_u] \tag{9d}$$

$$\boldsymbol{h}_c(\bar{\boldsymbol{\xi}}(\tau)) \leq 0 \tag{9e}$$

and for the TT-MPC as

$$\min_{\boldsymbol{e}_{tt}, \dot{\boldsymbol{e}}_{tt}, e_s, \boldsymbol{u}} \int_{t_k}^{t_k+T} J_{tt}(\tau, \bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{u}}(\tau)) \mathrm{d}\tau \tag{10a}$$

s.t.:

$$\dot{\bar{\boldsymbol{\chi}}}(\tau) = \boldsymbol{f_\chi}(\bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{u}}(\tau)) \tag{10b}$$

$$\bar{\boldsymbol{\chi}}(t_k) = \boldsymbol{\chi}(t_k) \tag{10c}$$

$$\bar{\boldsymbol{\chi}}(\tau) \in [\bar{\boldsymbol{\chi}}_l, \bar{\boldsymbol{\chi}}_u] \tag{10d}$$

$$\boldsymbol{h}_c(\bar{\boldsymbol{\chi}}(\tau)) \leq 0 \tag{10e}$$

where subscript $u$ refers to the upper bounds on the states, subscript $l$ refers to the lower bound, and $\boldsymbol{h}_c$ describes other constraints such as obstacles in the path. The bar denotes internal states of the MPC.

The cost integrands are defined as

$$\begin{aligned} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{pf}(\tau)^T \boldsymbol{Q} \bar{\boldsymbol{e}}_{pf} + \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{pf}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{pf} \\ &+ \frac{1}{2}\bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R} \bar{\boldsymbol{u}}(\tau) \\ &+ \frac{1}{2} q \bar{e}_s(\tau)^2 + \frac{1}{2} r \bar{v}(\tau)^2 \end{aligned} \tag{11}$$

for the MPFC and

$$\begin{aligned} J_{tt}(\tau, \bar{\boldsymbol{\chi}}(\tau), \bar{\boldsymbol{w}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{tt}(\tau)^T \boldsymbol{Q} \bar{\boldsymbol{e}}_{tt} + \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{tt}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{tt} \\ &+ \frac{1}{2}\bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R} \bar{\boldsymbol{u}}(\tau) \end{aligned} \tag{12}$$

for the TT-MPC. The matrices $\boldsymbol{Q}$, $\boldsymbol{Q}_d$, and $\boldsymbol{R}$ are positive definite. The scalars $q$ and $r$ are positive. We have included the derivative of the path deviation to reduce oscillations.

Solving the OCPs can be done by an indirect approach using Pontryagin's Maximum Principle as done in [9], or a direct approach as done in [7]. We will apply the direct simultaneous approach, reformulating the OCP as an NLP by discretizing the problem. The direct simultaneous approach is most common in real-time applications, with existing software support such as ACADO [6] and CasADI [10].

## C. Nonlinear Program and Interior Point

In this section we only give the discretization of (9) as the TT-MPC is similar and simpler. To discretize the OCP we use two different integration methods: the 4th order Runge-Kutta (RK4), and collocation based on Lagrange polynomials with $d$ Legendre points.

The control input is a piecewise continuous function, constant on intervals of length $\delta_t$, which is the length of our timesteps. This gives us a horizon of length $N_T = T/\delta_t$. With the simultaneous approach we use the integration method between each time step, and constrain the result and next state to be equal.

*1) Runge-Kutta:* Given a $\delta_t$, RK4 [11] gives us the equation

$$\bar{\boldsymbol{\xi}}_{k+1} = \bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{6}(k_1 + 2k_2 + 2k_3 + k_4) = F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \tag{13}$$

with

$$k_1 = \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k), \tag{14a}$$

$$k_2 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_1\frac{\delta_t}{2}, \bar{\boldsymbol{w}}_k\right), \tag{14b}$$

$$k_3 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_2\frac{\delta_t}{2}, \bar{\boldsymbol{w}}_k\right), \tag{14c}$$

$$k_4 = \boldsymbol{f_\xi}\left(\bar{\boldsymbol{\xi}}_k + k_3\delta_t, \bar{\boldsymbol{w}}_k\right). \tag{14d}$$

The resulting NLP is then

$$\min_{\boldsymbol{q}} \quad \phi(\boldsymbol{q}) \tag{15a}$$

$$\text{s.t.:}$$

$$\boldsymbol{f_e}(\boldsymbol{q}) = \boldsymbol{0} \tag{15b}$$

$$\boldsymbol{h_e}(\boldsymbol{q}) \le \boldsymbol{0}, \tag{15c}$$

where $\boldsymbol{q} = [\bar{\boldsymbol{\xi}}_k^T, \bar{\boldsymbol{w}}_k^T, \dots, \bar{\boldsymbol{\xi}}_{k+N_T-1}^T, \bar{\boldsymbol{w}}_{k+N_T-1}^T, \bar{\boldsymbol{\xi}}_{k+N_T}^T]^T$, the cost function is approximated with the rectangle method

$$\phi(\boldsymbol{q}) = \sum_{j=k}^{k+N_T} \delta_t J_{pf}(t_j, \bar{\boldsymbol{\xi}}_j, \bar{\boldsymbol{w}}_j), \tag{16}$$

and

$$\boldsymbol{f_e}(\boldsymbol{q}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1} - F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - F(\bar{\boldsymbol{\xi}}_{k+N_T-1}, \bar{\boldsymbol{w}}_{k+N_T-1}) \end{bmatrix}. \tag{17}$$

The inequality constraints use (9d)-(9e) enforced on $\bar{\boldsymbol{\xi}}_i$ for $i = k, \dots, k + N_t$.

*2) Collocation:* For the collocation method, with $d$ collocation points, we define $j = 0, \dots, d$ Lagrange polynomials

$$L_j(\tilde{\tau}) = \prod_{r=0, r\neq j}^{d} \frac{\tilde{\tau} - \theta_r}{\theta_j - \theta_r} \tag{18}$$

where $\tilde{\tau} \in [0,1]$, $\theta_0$ is 0, and the other $\theta_i$ are Legendre collocation points. The approximation of the state trajectory between $t_k$ and $t_{k+1}$ is then

$$\bar{\boldsymbol{\xi}}(\tau) = \sum_{j=0}^{d} L_j\left(\frac{\tau - t_k}{\delta_t}\right)\bar{\boldsymbol{\xi}}_{k,j}, \text{ for } \tau \in [t_k, t_{k+1}] \tag{19}$$

where $\bar{\boldsymbol{\xi}}_{k,j}$ are optimization variables included in $\boldsymbol{q}$. Requiring the simultaneous constraint to hold, and the derivatives to fit on the Legendre points, we have

$$\bar{\boldsymbol{\xi}}_{k+1,0} = \sum_{j=0}^{d} L_j(1)\bar{\boldsymbol{\xi}}_{k,j} \tag{20}$$

$$\boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,j}, \bar{\boldsymbol{w}}_k) - \frac{1}{\delta_t}\sum_{r=0}^{d} \dot{L}_r(\theta_j) = 0, \text{ for } j = 1, \dots, d \tag{21}$$

where $L_r(1)$ and $\dot{L}_r(\theta_j)$ are independent of $t_k$ and are precomputed. This gives a similar structure to (15) with the cost function evaluated with $\bar{\boldsymbol{\xi}}_{i,0}$ for $i = k, \dots, k + N_t$. We have chosen to evaluate (9d) at all states $\bar{\boldsymbol{\xi}}_{k,j}$ and the nonlinear inequality constraints, (9e), at $\bar{\boldsymbol{\xi}}_{k,0}$. This reduces the computational burden, but allows the collocation points $\bar{\boldsymbol{\xi}}_{k,j}$ between $t_k$ and $t_{k+1}$ to violate the nonlinear inequality constraints. The optimization vector is of the form

$$\begin{aligned}\boldsymbol{q} =& [\bar{\boldsymbol{\xi}}_{k,0}^T, \bar{\boldsymbol{\xi}}_{k,1}^T, \dots, \bar{\boldsymbol{\xi}}_{k,d}^T, \bar{\boldsymbol{w}}_k^T, \\ & \dots, \bar{\boldsymbol{\xi}}_{k+N_T-1,d}^T, \bar{\boldsymbol{w}}_{k+N_T-1}^T, \\ & \bar{\boldsymbol{\xi}}_{k+N_T,0}^T, \dots, \bar{\boldsymbol{\xi}}_{k+N_T,d}^T]^T\end{aligned} \tag{22}$$

and equality constraint function

$$\boldsymbol{f_e}(\boldsymbol{q}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_{k,0} - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1,0} - \sum_{r=0}^{d} L_r(1)\bar{\boldsymbol{\xi}}_{k,r} \\ \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,0}, \bar{\boldsymbol{w}}_k) - \sum_{r=0}^{d} \dot{L}_r(\theta_0)\bar{\boldsymbol{\xi}}_{k,0} \\ \vdots \\ \boldsymbol{f_\xi}(\bar{\boldsymbol{\xi}}_{k,1}, \bar{\boldsymbol{w}}_k) - \sum_{r=0}^{d} \dot{L}_r(\theta_1)\bar{\boldsymbol{\xi}}_{k,d} \\ \vdots \\ \end{bmatrix}. \tag{23}$$

*3) Interior point solver:* Primal Interior point methods consider NLPs of the form

$$\min_{\tilde{\boldsymbol{q}}} \quad \phi(\tilde{\boldsymbol{q}}) - \mu\sum_{i=0}^{\tilde{n}} \ln(\tilde{\boldsymbol{q}}_n) \tag{24a}$$

$$\text{s.t.:}$$

$$\boldsymbol{f_e}(\tilde{\boldsymbol{q}}) = \boldsymbol{0} \tag{24b}$$

where $\tilde{\boldsymbol{q}}$ includes slack variables to make $\boldsymbol{h_e}$ an equality constraint, and $\mu$ defines the steepness of the barrier associated with the slack variables. For large values of $\mu$ the $\ln$ term dominates and the solution tends to the center of the feasible region. As $\mu$ decreases, $\phi$ dominates and the solution moves towards the optimal solution. Solving for decreasing $\mu$ will converge to the solution of (15). Interior point methods are difficult to warm-start, as a too low $\mu$ may make certain slack variables prematurely small and cause slow convergence. In the timing tests, we have not used warm start as the initial states are random, but in the MPC implementation we use the previous $\bar{\boldsymbol{\xi}}$ predictions as an initial guess.

We will use the interior point solver IPOPT [12], a primal-dual interior point solver, solving (24) using the primal-dual equations, see section 3.1 in [12]. Interior point methods have consistent runtime with respect to problem size, allowing us to potentially include more states with little effect on runtime.

Convergence of the MPFC is ensured using terminal sets and penalties as constructed in [4] where an example is given for the same system as ours with different parameters. In this article we do not consider the terminal cost and penalty.

## III. Simulation

We consider a two-link manipulator. This can easily be extended to 6 degrees-of-freedom, and results here are indicative of the larger systems as interior point methods are consistent with respect to the number of variables. The system was implemented using Python and the CasADi framework [10]. CasADi allows us to define symbolic expressions for the various equations in (9), and evaluate the derivatives using algorithmic differentiation, e.g. for RK4, which may be difficult to do by hand. The framework supports IPOPT [12].

## *A. System*

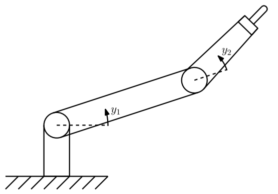


Fig. 1: Two-link manipulator with 2 revolute joints.

The robot has two links of length $L_1$ and $L_2$, with link masses $M_1$ and $M_2$ at $L_{c1}$ and $L_{c2}$, and masses $m_1$ and $m_2$ at the joints. The point of interest is the tip of the end effector. The system is described by

$$\dot{\boldsymbol{y}} = \dot{\boldsymbol{x}} \tag{25a}$$

$$\dot{\boldsymbol{x}} = \boldsymbol{M}(\boldsymbol{y})^{-1}\left(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{y},\boldsymbol{x})\boldsymbol{x} - \boldsymbol{G}(\boldsymbol{y})\right) \tag{25b}$$

with

$$\boldsymbol{M}(\boldsymbol{y}) = \begin{bmatrix} a_1 + a_2\cos(y_2) & \frac{1}{2}a_2\cos(y_2) + a_3 \\ \frac{1}{2}a_2\cos(y_2) + a_3 & a_3 \end{bmatrix} \tag{26}$$

$$\boldsymbol{C}(\boldsymbol{y},\boldsymbol{x}) = \begin{bmatrix} -\frac{1}{2}a_2\sin(y_2)x_2 & -\frac{1}{2}\sin(y_2)(x_1+x_2) \\ \frac{1}{2}a_2\sin(y_2)x_1 & 0 \end{bmatrix} \tag{27}$$

$$\boldsymbol{G}(\boldsymbol{y}) = \begin{bmatrix} g_1\cos(y_1) + g_2\cos(y_1+y_2) \\ g_2\cos(y_1+y_2) \end{bmatrix}, \tag{28}$$

where

$$a_1 = I_1 + I_2 + m_1L_{c1}^2 + m_2(L_1^2 + L_2^2), \tag{29}$$

$$a_2 = 2m_2L_{c2}L_1, \quad a_3 = m_2L_{c2}^2 + I_2, \tag{30}$$

$$g_1 = (L_{c1}(m_1+M_1) + L_1(m_2+M_2))g, \tag{31}$$

$$g_2 = L_{c2}(m_2+M_2)g. \tag{32}$$

For brevity we give $a_1, a_2, a_3, g_1, g_2 \in \mathbb{R}$ in Table I, for more information see [13]. The joint angles are defined as in Fig.1. The maximum torques are 30 Nm, the timesteps are $\delta_t = 0.01$ s, and if not otherwise specified the horizon is $T = 0.20$ s.

TABLE I: System parameters

| Parameter | $a_1$ | $a_2$ | $a_3$ | $g_1$ | $g_2$ |
|---|---|---|---|---|---|
| Value | 0.5578 | 0.2263 | 0.0785 | 17.0694 | 4.3164 |

TABLE II: MPC Parameters

| Parameter | $\boldsymbol{Q}$ | $\boldsymbol{Q}_d$ | $\boldsymbol{R}$ | $q$ | $r$ |
|---|---|---|---|---|---|
| MPFC | $10^4\boldsymbol{I}_{2\times2}$ | $10^1\boldsymbol{I}_{2\times2}$ | $10^{-3}\boldsymbol{I}_{2\times2}$ | 1 | $10^{-3}$ |
| TT-MPC | $10^4\boldsymbol{I}_{2\times2}$ | $10^1\boldsymbol{I}_{2\times2}$ | $10^{-3}\boldsymbol{I}_{2\times2}$ | | |

In order to study obstacle avoidance, we include obstacles $o_i$ as bounding circles with known radius $r_{o_i}$ and position $\boldsymbol{p}_{o_i}$. Their inequality equations are

$$\boldsymbol{h}_{o_i} = \left\|\boldsymbol{f_p}(\bar{\boldsymbol{\xi}}(t)) - \boldsymbol{p}_{o_i}\right\|^2 - r_{o_i}^2 > 0. \tag{33}$$

In actual applications a vision system would bound detected objects or people by a circle that the point of interest is not to enter. When present we consider two obstacles with: $r_{o_1} = 0.02$m, at $p_{o_1} = [0.55, 0.75]^T$, and $r_{o_2} = 0.04$m, at $p_{o_2} = [0.4, 0.4]^T$.

The reference path is a circle of radius 0.2 with center at $[0.55, 0.55]^T$.

## *B. Results*

*1) Moving to origin:* In Fig.2 we see the Cartesian paths of the Runge-Kutta TT-MPC and MPFC. The black dot is $\varrho(0)$. For this simulation we used maximum joint speeds of $0.5\pi$ rad/s to exaggerate the differences. The set-point of the TT-MPC moves gradually while the TT-MPC is approaching the path. The MPFC on the other hand first approaches the origin of the path, then moves along it. If $q$ is large compared to $\boldsymbol{Q}$, we will move along the path faster than to the path. The MPFC has come further with no difference in path deviation as $\dot{s}$ is greater than the rate of $t$, allowing it to move faster along the path.

*2) Obstacles:* In Fig.3 two obstacles have been placed in the path, and the speed constraints are removed. Both MPFC and TT-MPC pass the first obstacle, but the MPFC stops at the second. The second obstacle is too large, and the horizon is not long enough to pass behind the obstacle. The MPFC will decrease $\dot{s}$ to zero, see $t \approx 2$s in Fig.5, whereas the TT-MPC will have a gradually increasing cost as the trajectory set-point moves forward through the obstacle, forcing it around the object.

When obstructed, there was a difference between the two integration methods for the TT-MPC. We saw that the collocation method left the TT-MPC path closer to the obstacles, see Fig.4. The MPFC decreased $\dot{s}$ upon approaching the first object, at $t \approx 0.9$s in Fig.5, and was not affected by integration differences.

*3) Timing:* The simulations were performed on a Macbook Pro with a 2.5 Ghz i7 CPU. Using the compilation feature of CasADi we can create implementations that approach speeds needed for real-time systems. To compare the timings of the two integration methods we have performed a Monte-Carlo simulation of the MPFC with uniformly distributed initial positions in the upper right quadrant of the workspace. Box plot of the solver using RK4 for varying horizon lengths is given in Fig.6. In Fig.7 we give the same for the collocation method.

CasADi gives timing statistics of the solver. Upon inspection it appears that the collocation method has faster evaluation of the constraint functions, Hessian of the problem Lagrangian, and generally fewer iterations, but the increased optimization vector length makes the solver slower. In Table III we give typical timings of the solver. The cost function and cost gradient are not included as they were the same and approximately 1 ms.

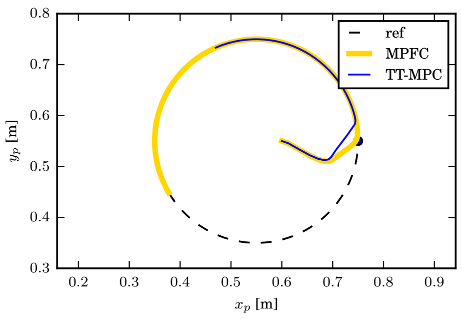


Fig. 2: TT-MPC and MPFC moving from the same start point towards the path. The blue dot is the start of the reference path.

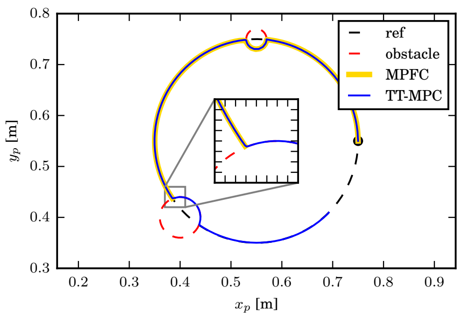


Fig. 3: TT-MPC and MPFC initialized close to the path origin. The path is obstructed by a small and a large object. Both controllers pass the first object, but the MPFC does not pass the second.

TABLE III: Typical timing statistics T = 0.6

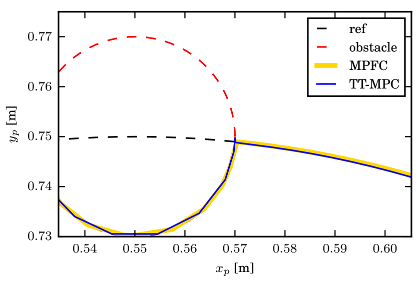
| | Constr.[1] | $\nabla_{\boldsymbol{q}}$Constr. | $\boldsymbol{H}(\mathcal{L})^2$ | Iter. | Solver tot. |
|---|---|---|---|---|---|
| MPFC-RK4 | 0.03 ms | 0.16 ms | 0.52 ms | 27 | 0.045 s |
| MPFC-Col | 0.03 ms | 0.08 ms | 0.15 ms | 24 | 0.102 s |
| TT-MPC-RK4 | 0.03 ms | 0.23 ms | 0.66 ms | 14 | 0.039 s |
| TT-MPC-Col | 0.03 ms | 0.07 ms | 0.11 ms | 14 | 0.046 s |

## IV. Discussion and Future work

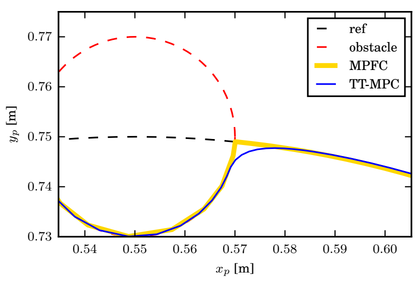
The collocation method has a sparse structure in the equality constraints, and relies on evaluation of $\boldsymbol{f}_{\boldsymbol{\xi}}$, whereas the RK4 requires evaluation of its algorithimic differentiation.

[1]The constraint function include both the inequality constraints and the ODE dynamics.

[2]Hessian of the problem Lagrangian.



(a) Collocation method



(b) RK4

Fig. 4: Difference between the two integration methods when the path is obstructed. Only evident in TT-MPC as the MPFC slows sufficiently down before the obstacle to not encounter the integration problems.

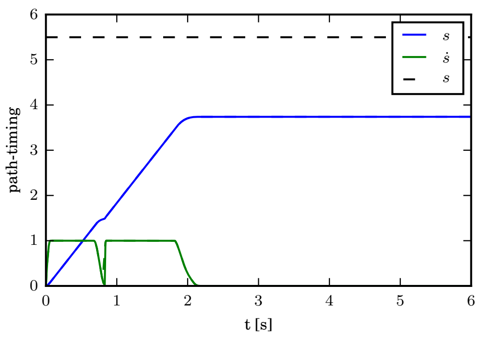


Fig. 5: Timing parameter $s$, blue line, and $\dot{s}$, green line, when MPFC follows the obstructed path. First object encountered at $t \approx 0.9$s, second object encountered at $t \approx 2$s.

This gives the collocation method faster function evaluations, but it did not appear to be sufficient to make the collocation method faster than RK4. The solver itself took more time with the increased number of states. For systems with complex dynamics the collocation method may be necessary. TT-MPC had fewer states, simpler dynamics and was faster

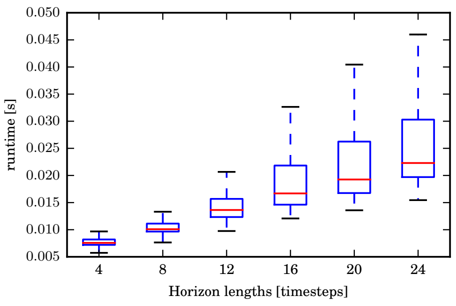


Fig. 6: Boxplot of 2000 RK4 MPFC solver run times.

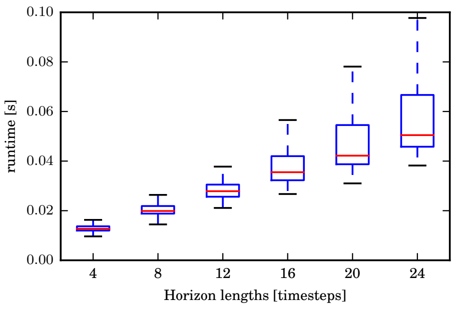


Fig. 7: Boxplot of 2000 collocation MPFC solver run times.

on the whole, with the same observations as for the MPFC regarding integration method.

The MPFC has the ability to stop along its path. For time-critical systems this is not desired, but for robots in open environments it can prove useful. It also suggests that when obstructed by an unknown object, it may push against the object with a constant force. In future experiments this will be investigated further. The TT-MPC will observe a growing difference in the current position and the desired position. With a known obstruction it will project the path onto the constraint attempting to minimize the path error. Suddenly removing the obstruction should result in the TT-MPC moving towards its current set point as fast as possible. With an unknown obstruction the TT-MPC may exert a gradually increasing force on the obstruction.

For the MPC to be real-time feasible, we require the solver to run faster than the control interval used. In this article we have considered a control interval of length $\delta_t = 0.01$ s. For low horizon lengths we are approaching such timing with the CasADi running in Python. Future work will extend this framework for a 6 degrees-of-freedom robot with a 3D path. The low horizon length needed to be able to achieve fast run time of the solver suggests that terminal constraints may be required in the final system.

The obstructions considered in this article were static and known apriori. Future work may include varying number of obstacles that enter the robot workspace.

## V. Conclusion

The model predictive path-following controller gives rise to a set of new design opportunities. Of most value for obstructed environments is the fact that it may freely stop and resume along its path. The question is whether a constraint ends the path, as the path-following controller did, or whether the robot should move along the path projected onto the constraint, as the trajectory tracking controller did.

We also saw that the interior point method of IPOPT interfaced through CasADi in Python, approached timings we would desire in a real-time systems.

## VI. Acknowledgements

The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

## References

[1] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, “Time-optimal path tracking for robots: A convex optimization approach,” *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[2] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. Swevers, and J. De Schutter, “Convex time-optimal robot path following with Cartesian acceleration and inertial force and torque constraints,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 227, no. 10, pp. 724–732, nov 2013.

[3] T. Faulwasser, B. Kern, and R. Findeisen, “Model predictive path-following for constrained nonlinear systems,” *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, no. 3, pp. 8642–8647, 2009.

[4] T. Faulwasser and R. Findeisen, “Nonlinear Model Predictive Control for Constrained Output Path Following,” *IEEE Transactions on Automatic Control*, vol. 9286, no. c, pp. 1–1, 2016.

[5] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, “Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot,” *IEEE Transactions on Control Systems Technology*, pp. 1–7, 2016.

[6] B. Houska, H. J. Ferreau, and M. Diehl, “ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization,” *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] D. Lam, C. Manzie, and M. Good, “Model predictive contouring control,” in *49th IEEE Conference on Decision and Control (CDC)*, vol. 86, no. 8. IEEE, dec 2010, pp. 6137–6142.

[8] ——, “Application of model predictive contouring control to an X-Y table,” *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, pp. 10 325–10 330, 2011.

[9] M. Böck and A. Kugi, “Real-time nonlinear model predictive path-following control of a laboratory tower crane,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2014.

[10] J. Andersson, “A General-Purpose Software Framework for Dynamic Optimization,” PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.

[11] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics, 2003.

[12] A. Wächter and L. T. Biegler, *On the Implementation of Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*, 2006, vol. 106, no. 1.

[13] A. Astolfi, D. Karagiannis, and R. Ortega, *Nonlinear and Adaptive Control with Applications (Communications and Control Engineering)*, 2007.

# Paper 3 Mid-Level MPC and 6 DOF Output Path Following for Robotic Manipulators

M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, “Mid-Level MPC and 6 DOF output path following for robotic manipulators,” *IEEE Conference on Control Technology and Applications (CCTA)*, Mauna Lani, HI, 2017, pp. 450-456.

# Mid-Level MPC and 6 DOF Output Path Following for Robotic Manipulators

Mathias Hauan Arbo*, Esten Ingar Grøtli† and Jan Tommy Gravdahl*
*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
†Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

***Abstract*—In this article we discuss some of the benefits of using an MPC as a mid-level controller between the path generator and the low-level joint controller of a robot system. The MPC handles rudimentary runtime constraints that are not considered during path generation. We compare two task space oriented controllers: the model predictive path following controller and the model predictive trajectory tracking controller. We describe a 6 degrees of freedom reference path in terms of three points, and use this to experimentally verify the results with a UR5 robot and a UR3 robot.**

## I. Introduction

Articulated industrial robots are used for a large variety of tasks which often entail moving the end-effector along a precomputed Cartesian path. When in enclosed spaces we can ensure that objects and obstructions are exactly known, but the real world is not so orderly. Objects may be moved around, obstructing the task at hand, motivating the use of controllers that can explicitly handle these observable constraints the obstructions impose. For some tasks obstructions can be handled during path generation, but when the path itself is pertinent, the obstructions should be considered during path execution.

The nonlinear model predictive controller (NMPC) views the control objective as an optimal control problem (OCP) with constraints and dynamics. Based on the predicted behavior of the system, it optimizes the input sequence so as to minimize an objective function. The first input in this sequence is applied to the actual system and the OCP is reformulated with new initial conditions based on the results of applying the input to the system. NMPCs allow us to handle rudimentary obstructions during path execution as constraints on the OCP.

There are two common schools in control of industrial manipulators. One considers the dynamics of the robot, where the torque applied at each joint as controllable. The other uses servo control on the robot joints, handling gravitational, Coriolis, friction and other forces internally, and exposes the end-user to kinematic references of the low-level controller. Kinematic control uses either position, velocity, or acceleration setpoints. The benefit of the kinematic approach is that it is simple and intuitive for end-users during free moving tasks. For fast-moving, high-inertia, or interaction tasks, dynamics play a greater role. In this article we will use joint velocity setpoints as the control interface, as this allows for application to different manipulators by redefining the forward kinematics.

We consider two MPC approaches: the model predictive path following controller (MPFC, following the convention of [1]), and the model predictive trajectory tracking controller (MPTTC). With MPTTC, the robot is to follow a path with an explicit path-timing. In [2], the path-timing is formulated as an OCP with constraints on the states. This is an open-loop approach that predefines a path-timing law for the trajectory tracking controller. In [3] this was extended to include constraints on the acceleration and inertial forces at the end-effector. For the MPFC, the path-timing dynamics are a part of the MPC, and are handled closed-loop. This allows the robot to move to minimize the deviation from the path, before moving along the path as will be demonstrated.

We will consider the output-path following MPFC of Faulwasser et al. [1]. This formulation focuses on paths defined in the output space. In [4], the MPFC is shown to converge to the path given appropriately chosen terminal constraints and penalties. In [5] the MPFC is used to generate torque inputs for a KUKA LWR IV robot. The OCP is solved using the ACADO framework [6], which uses sequential programming and the qpOASES active set solver. The robot is constrained to act as a two-link planar arm, and the path has 2 degrees-of-freedom (DOF).

The MPFC formulation is implemented for real-time contouring control of an x-y table in [7], where current commands are used to specify torques on the two servo drives. The dynamics are linearised and the OCP is formulated as a quadratic program which is solved using an active set solver. In [8] it is implemented for control of a toy tower crane. The tower crane is controlled with acceleration setpoints, and the OCP is solved using the gradient projection method which uses Pontryagin's maximum principle to solve the analytical OCP.

This article is a continuation of [9] which looks at the MPFC and MPTTC for a 2 DOF double pendulum system. It notes that Runge-Kutta gives a faster solver than collocation if the system is sufficiently simple, and shows how the MPFC can stop at obstructions that are not profitable, from the OCP perspective, to pass around. The MPTTC on the other hand will move along the nullspace of the constraint to follow the path.

In this article we extend the results of [5] and [9] to 6 DOF paths using a novel method of following 3 points, and argue for the flexibility of output path following systems with kinematic control. We also provide experimental results of the 6 DOF path formulation for a spiral path with MPTTC and MPFC, and exemplify the flexibility by following a 3D Lissajous path with the UR3 robot and the UR5 robot.

## II. Theory

### A. Control

In Fig.1 we show a possible realization of the total control system. At the highest level a geometric path is precomputed based on the task to be performed, only considering static obstructions or task related obstructions. When the robot is to execute the task, the path is given to the mid-level MPC which uses cameras or similar systems to identify obstructions. The MPC gives kinematic setpoints to the fast low-level joint controllers on the robot. The output-path oriented control of the MPC means that we only need to generate the new forward kinematics for using it with a new robot, given that the interface to the low-level controller is similar. This flexible design means that tasks can be shared between different robot setups without the need for redesigning the path following and obstruction handling portion of the controller. The tasks can also be defined independent of the robots to be used, and a system can be devised which distributes ones available robots to the appropriate tasks.

### B. Robot and Path

We consider a 6 DOF articulated robot, with $\boldsymbol{q} \in \mathbb{R}^6$ joint coordinates, and angular velocity setpoints $\boldsymbol{u} \in \mathbb{R}^6$.

*Assumption* 1. The low-level controllers are assumed to be sufficiently fast for

$$\dot{\boldsymbol{q}}(t) = \boldsymbol{u}(t) \tag{1}$$

to represent the robot dynamics.

Joint constraints are enforced as $\boldsymbol{q}(t) \in [\boldsymbol{q}_l, \boldsymbol{q}_u]$, and joint velocities are $\boldsymbol{u}(t) \in [\boldsymbol{u}_l, \boldsymbol{u}_u]$.

The base frame is located at the base of the robot, and the robot has known forward kinematics described using the Denavit-Hartenberg convention. The rotation from a reference frame situated at joint $i$ to the base frame $b$ is $\boldsymbol{R}_i^b \in \mathbb{R}^{3\times3}$ for $i = 1, \dots, 6$, and $\boldsymbol{p}_{bi}^b$ is the coordinates of the reference frame $i$ relative to frame $b$ expressed in terms of frame $b$. The 6 DOF path is defined as $\boldsymbol{R}_d^b(s)$ and $\boldsymbol{p}_{bd}^b(s)$ defining rotation of the

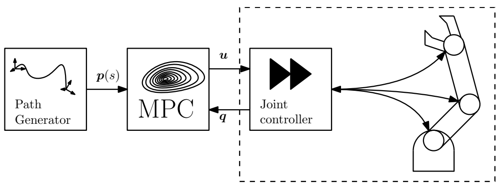


Fig. 1: Control hierarchy described in this article. The portion in the dashed box can be quickly changed if $\boldsymbol{u}$ and $\boldsymbol{q}$ are available through the same interface.

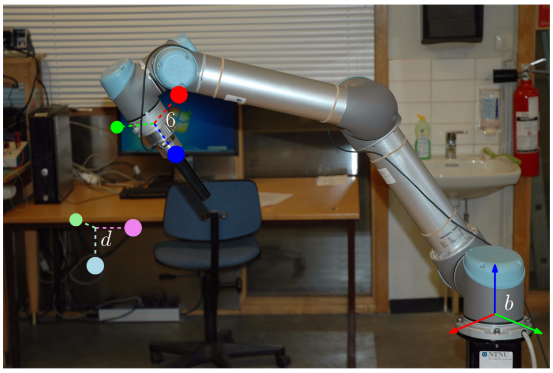


Fig. 2: The red, green, and blue points, attached to the UR5 follow the path of their lighter counterparts.

desired reference frame $d$ w.r.t. the base frame, and $s$ being the path-timing variable. We want the end-effector frame 6 to follow the desired reference frame $d$.

A variety of methods exist for representing rotations, e.g. quaternions, Euler angles, etc. We propose to use the intuitive method of defining three desired paths corresponding to orthogonal vectors from the desired reference frame:

$$\boldsymbol{p}(s) = \begin{bmatrix} \boldsymbol{p}_1(s) \\ \boldsymbol{p}_2(s) \\ \boldsymbol{p}_3(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_d^b(s(t))[1,0,0]^T + \boldsymbol{p}_{bd}^b(s(t)) \\ \boldsymbol{R}_d^b(s(t))[0,1,0]^T + \boldsymbol{p}_{bd}^b(s(t)) \\ \boldsymbol{R}_d^b(s(t))[0,0,1]^T + \boldsymbol{p}_{bd}^b(s(t)) \end{bmatrix} \tag{2}$$

which three points in the end-effector frame are to follow. The points in the base frame are found using the forward kinematics as

$$\boldsymbol{h}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{q}) \\ \boldsymbol{h}_2(\boldsymbol{q}) \\ \boldsymbol{h}_3(\boldsymbol{q}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{b6}^b(\boldsymbol{q})[1,0,0]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \\ \boldsymbol{R}_{b6}^b(\boldsymbol{q})[0,1,0]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \\ \boldsymbol{R}_{b6}^b(\boldsymbol{q})[0,0,1]^T + \boldsymbol{p}_{b6}^b(\boldsymbol{q}(t)) \end{bmatrix}, \tag{3}$$

where $\boldsymbol{h}_i(\boldsymbol{q})$ is the forward kinematics to point $i$.

This is visualized in Fig.2 as the red, green, and blue dots at the end-effector moving to the desired positions at their lighter counterparts.

We define the deviation from path as

$$\boldsymbol{e}_{\boldsymbol{p}}(t) := \boldsymbol{h}(\boldsymbol{q}(t)) - \boldsymbol{p}(s(t)) \tag{4}$$

with $\boldsymbol{e}_{\boldsymbol{p}}(t) \in \mathbb{R}^9$.

For the MPFC, we must also define the path-timing dynamics. We will use a simple double integrator

$$\ddot{s}(t) = v(t) \tag{5}$$

where $v(t) \in [-\infty, \infty]$ is the input, and $\dot{s}(t) \in [0, \dot{s}_u]$ is the non-negative path speed to ensure forward motion along the path. For more information on choice of path-timing dynamics we refer the reader to [4] and [8]. In theory and simulations we only required a single integrator, but in the experiments the double integrator path-dynamics performed better. We believe this was due to the delay caused by the solver making the system overshoot its desired path timing.

We will consider paths with a specific start and finish, $s \in [0, s_f]$ where $s_f$ denotes the final value. We also define the deviation from the final value as

$$e_s(t) := s(t) - s_f. \tag{6}$$

For the MPFC we define the augmented state vector $\boldsymbol{\xi} := [\boldsymbol{q}, s, \dot{s}]^T$, augmented input $\boldsymbol{w} := [\boldsymbol{u}, 0, v]^T$, augmented deviation $\boldsymbol{e}_{\boldsymbol{\xi}} = [\boldsymbol{e}_{\boldsymbol{p}}^T, e_s]^T$, and augmented system

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{A}\boldsymbol{\xi}(t) + \boldsymbol{w}(t) \tag{7}$$

with

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0}_{6\times6} & \boldsymbol{0}_{6\times1} & \boldsymbol{0}_{6\times1} \\ \boldsymbol{0}_{1\times6} & 0 & 1 \\ \boldsymbol{0}_{1\times6} & 0 & 0 \end{bmatrix}. \tag{8}$$

### C. Optimal Control Problem

Using the previously defined dynamics and deviations, we describe the OCP for the MPFC as

$$\begin{aligned}
&\min_{\boldsymbol{e}_{\boldsymbol{\xi}}, \dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{w}} \int_{t_k}^{t_k+T} J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) \mathrm{d}\tau && \text{(9a)}\\
&\text{s.t.:}\\
&\quad \dot{\bar{\boldsymbol{\xi}}}(\tau) = \boldsymbol{A}\bar{\boldsymbol{\xi}}(\tau) + \bar{\boldsymbol{w}}(\tau) && \text{(9b)}\\
&\quad \bar{\boldsymbol{\xi}}(t_k) = \boldsymbol{\xi}(t_k) && \text{(9c)}\\
&\quad \bar{\boldsymbol{\xi}}(\tau) \in [\bar{\boldsymbol{\xi}}_l, \bar{\boldsymbol{\xi}}_u] && \text{(9d)}\\
&\quad \bar{\boldsymbol{w}}(\tau) \in [\bar{\boldsymbol{w}}_l, \bar{\boldsymbol{w}}_u] && \text{(9e)}\\
&\quad \boldsymbol{h}_c(\bar{\boldsymbol{\xi}}(\tau)) \leq 0 && \text{(9f)}
\end{aligned}$$

and for the MPTTC as

$$\begin{aligned}
&\min_{\boldsymbol{e}_{\boldsymbol{p}}, \dot{\boldsymbol{e}}_{\boldsymbol{p}}, \boldsymbol{u}} \int_{t_k}^{t_k+T} J_{tt}(\tau, \bar{\boldsymbol{q}}(\tau), \bar{\boldsymbol{u}}(\tau)) \mathrm{d}\tau && \text{(10a)}\\
&\text{s.t.:}\\
&\quad \dot{\bar{\boldsymbol{q}}}(\tau) = \bar{\boldsymbol{u}}(\tau) && \text{(10b)}\\
&\quad \bar{\boldsymbol{q}}(t_k) = \boldsymbol{q}(t_k) && \text{(10c)}\\
&\quad \bar{\boldsymbol{q}}(\tau) \in [\bar{\boldsymbol{q}}_l, \bar{\boldsymbol{q}}_u] && \text{(10d)}\\
&\quad \bar{\boldsymbol{u}}(\tau) \in [\bar{\boldsymbol{u}}_l, \bar{\boldsymbol{u}}_u] && \text{(10e)}\\
&\quad \boldsymbol{h}_c(\bar{\boldsymbol{q}}(\tau) \leq 0 && \text{(10f)}
\end{aligned}$$

where the bar is to signify that these are internal states of the OCP, subscript $u$ refers to the upper bounds, and subscript $l$ refers to the lower bounds. The function $\boldsymbol{h}_c$ defines other constraints such as obstructions, the end-effector remaining in the workspace, etc. The OCP uses samples from time $t_k$ and has a prediction horizon of length $T$.

The cost integrands are defined as

$$\begin{aligned}
J_{pf}(\tau, \bar{\boldsymbol{\xi}}(\tau), \bar{\boldsymbol{w}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{\boldsymbol{\xi}}(\tau)^T \boldsymbol{Q}_{\boldsymbol{\xi}} \bar{\boldsymbol{e}}_{\boldsymbol{\xi}}(\tau)\\
&+ \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)\\
&+ \frac{1}{2}\bar{\boldsymbol{w}}(\tau)^T \boldsymbol{R}_{\boldsymbol{w}} \bar{\boldsymbol{w}}(\tau)
\end{aligned} \tag{11}$$

fort the MPFC, and

$$\begin{aligned}
J_{tt}(\tau, \bar{\boldsymbol{q}}(\tau), \bar{\boldsymbol{u}}(\tau)) =& \frac{1}{2}\bar{\boldsymbol{e}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_{\boldsymbol{p}} \bar{\boldsymbol{e}}_{\boldsymbol{p}}(\tau)\\
&+ \frac{1}{2}\dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)^T \boldsymbol{Q}_d \dot{\bar{\boldsymbol{e}}}_{\boldsymbol{p}}(\tau)\\
&+ \frac{1}{2}\bar{\boldsymbol{u}}(\tau)^T \boldsymbol{R}_{\boldsymbol{u}} \bar{\boldsymbol{u}}(\tau)
\end{aligned} \tag{12}$$

for the MPTTC. We have $\boldsymbol{Q}_{\boldsymbol{\xi}} = \operatorname{diag}(\boldsymbol{Q}_{\boldsymbol{p}}, q_s)$ and $\boldsymbol{R}_{\boldsymbol{w}} = \operatorname{diag}(\boldsymbol{R}_{\boldsymbol{u}}, r_v)$ with $\boldsymbol{Q}_{\boldsymbol{p}}$, $\boldsymbol{Q}_d$ and $\boldsymbol{R}_{\boldsymbol{u}}$ being positive definite, and scalars $q_s$ and $r_v$ positive.

### D. Nonlinear Program

In this section we only give the discretization of (9) as the MPTTC is similar and simpler. We consider $\boldsymbol{u}(t)$ and $\boldsymbol{v}(t)$ to be piecewise continuous with time intervals of $\delta_t$. The prediction horizon has $N_T = T/\delta_t$ intervals. This means that the prediction horizon is discretized from step $t_k$ to $t_{k+N_T}$. Runge-Kutta of the 4th order (RK4) [10] gives

$$\bar{\boldsymbol{\xi}}_{k+1} = \bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) := \boldsymbol{F}(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \tag{13}$$

with

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{A}\bar{\boldsymbol{\xi}}_k + \bar{\boldsymbol{w}}_k, && \text{(14a)}\\
\boldsymbol{k}_2 &= \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{2}\boldsymbol{k}_1\right) + \bar{\boldsymbol{w}}_k, && \text{(14b)}\\
\boldsymbol{k}_3 &= \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \frac{\delta_t}{2}\boldsymbol{k}_2\right) + \bar{\boldsymbol{w}}_k, && \text{(14c)}\\
\boldsymbol{k}_4 &= \boldsymbol{A}\left(\bar{\boldsymbol{\xi}}_k + \delta_t\boldsymbol{k}_3\right) + \bar{\boldsymbol{w}}_k. && \text{(14d)}
\end{aligned}$$

We employ the simultaneous approach, and define the optimization vector as

$$\boldsymbol{x} = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k^T & \bar{\boldsymbol{w}}_k^T & \cdots & \bar{\boldsymbol{\xi}}_{k+N_T-1}^T & \bar{\boldsymbol{w}}_k^T & \bar{\boldsymbol{\xi}}_{k+N_T}^T \end{bmatrix} \tag{15}$$

where subscript $k$ means that it is the discretised value of the state at time $t_k$. The dynamics and initial value are accounted for by

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \boldsymbol{\xi}(t_k) \\ \bar{\boldsymbol{\xi}}_{k+1} - F(\bar{\boldsymbol{\xi}}_k, \bar{\boldsymbol{w}}_k) \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - F(\bar{\boldsymbol{\xi}}_{k+N_T-1}, \bar{\boldsymbol{w}}_{k+N_T-1}) \end{bmatrix} \tag{16}$$

and the constraints are accounted for by

$$\boldsymbol{f}_c(\boldsymbol{x}) = \begin{bmatrix} \bar{\boldsymbol{\xi}}_k - \bar{\boldsymbol{\xi}}_u \\ \bar{\boldsymbol{w}}_k - \bar{\boldsymbol{w}}_u \\ \vdots \\ \bar{\boldsymbol{\xi}}_{k+N_T} - \bar{\boldsymbol{\xi}}_u \\ \bar{\boldsymbol{\xi}}_l - \bar{\boldsymbol{\xi}}_k \\ \bar{\boldsymbol{w}}_l - \bar{\boldsymbol{w}}_k \\ \vdots \\ \bar{\boldsymbol{\xi}}_l - \bar{\boldsymbol{\xi}}_{k+N_T} \\ \boldsymbol{f}_c(\bar{\boldsymbol{\xi}}_k) \\ \vdots \\ \boldsymbol{f}_c(\bar{\boldsymbol{\xi}}_{k+N_T}) \end{bmatrix} \tag{17}$$

The resulting nonlinear program (NLP) is then

$$\min_{\boldsymbol{x}} \quad \phi(\boldsymbol{x}) \tag{18a}$$

s.t.:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0} \tag{18b}$$

$$\boldsymbol{f}_c(\boldsymbol{x}) \leq \boldsymbol{0}, \tag{18c}$$

where the cost function is approximated with Euler's method

$$\phi(\boldsymbol{x}) = \sum_{j=k}^{k+N_T-1} \delta_t J_{pf}(t_j, \bar{\boldsymbol{\xi}}_j, \bar{\boldsymbol{w}}_j). \tag{19}$$

## E. *Interior point solver*

Primal Interior point methods consider NLPs of the form

$$\min_{\tilde{\boldsymbol{x}}} \quad \phi(\tilde{\boldsymbol{x}}) - \mu \sum_{i=j}^{\tilde{n}} \ln(\tilde{\boldsymbol{x}}_i) \tag{20a}$$

s.t.:

$$\boldsymbol{f}(\tilde{\boldsymbol{x}}) = \boldsymbol{0} \tag{20b}$$

where $\tilde{\boldsymbol{x}}_i$ for $i < j$ are the previous optimization variables and $i < j$ are slack variables to make $\boldsymbol{f}_c$ an equality constraint. $\mu$ defines the steepness of the barrier associated with the slack variables. For large values of $\mu$ the ln term will dominate and the solution will tend to the middle of the feasible region. As $\mu$ decreases, $\phi$ will dominate and the solution will move towards the optimal solution. Solving (20) for decreasing $\mu$ will converge to the solution of the actual NLP (18).

The motivation for interior point solvers is that they have consistent runtime with respect to problem size, allowing us to potentially include more states and constraints without adversely affecting the runtime. They are however difficult to warm-start, as too low $\mu$ may make certain slack variables prematurely small and cause slow convergence. It was observed that warm-starting with the previously solved $\boldsymbol{x}$ gave a small decrease in runtime.

We will use the interior point solver IPOPT [11], a primal-dual interior point solver, solving (20) using the primal-dual equations, see Section 3.1 in [11]. Convergence of the MPFC can be ensured with terminal sets and penalties as in [4]. In this article we focus on run time and do not create terminal sets and penalties.

## III. Experimental Results

In this section we describe the experiments performed. To compare the MPFC and MPTTC we use a UR5 that is to follow a spiral path. To illustrate the simplicity of using the same approach for different robots, we use a 3D Lissajous curve that is executed both by a UR3 and a UR5.

## A. *Implementation*

The system was implemented using Python and the CasADi framework [12]. CasADi is a symbolic framework for defining optimization problems. It allows for: algorithmic differentiation, exploiting sparsity of the problem, and compiling the symbolic functions to C++ for faster execution. The framework supports a variety of solvers, both commercial and open-source. As of writing the fastest and most common solver is IPOPT [11].

We used the compilation flag "O2" to optimize the resulting functions. The experiments were performed on a Macbook Pro with a 2.5 Ghz i7 CPU. The timestep used in the simulations is $\delta_t = 0.05$, corresponding to an update rate of 20 Hz. The horizon has $N_T = 5$ timesteps corresponding to 0.25 s.

The forward kinematics were found using the Denavit-Hartenberg parameters described in [13]. The tuning parameters are given in Table I, and were the same for both the UR5 and the UR3.

## B. *Spiral path*

In this section the reference path of the MPTTC and MPFC is a downward moving spiral path with a constant rotation of $\pi$ rad around the $y$-axis from the base frame to the end-effector frame. The coordinate of the desired frame is

$$\boldsymbol{p}_{bd}^b(s) = \begin{bmatrix} 0.155\cos(2s) + 0.477 \\ 0.155\sin(2s) - 0.239 \\ 0.219 - 0.05s \end{bmatrix} \tag{21}$$

giving

$$\boldsymbol{p}(s) = \begin{bmatrix} \boldsymbol{p}_1(s) \\ \boldsymbol{p}_2(s) \\ \boldsymbol{p}_3(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{bd}^b(s) - [1,0,0]^T \\ \boldsymbol{p}_{bd}^b(s) + [0,1,0]^T \\ \boldsymbol{p}_{bd}^b(s) - [0,0,1]^T \end{bmatrix} \tag{22}$$

and the paths terminate at $s_f = 2\pi$. For the MPTTC we scale $s$ by 0.0125 so that a full rotation is completed after approximately 80 s.

TABLE I: MPC Parameters

| Parameter | $\boldsymbol{Q}_p$ | $\boldsymbol{Q}_d$ | $\boldsymbol{R}_u$ | $q_s$ | $r_v$ |
|---|---|---|---|---|---|
| MPFC | $10^7\boldsymbol{I}_{9\times9}$ | $1.5\cdot10^5\boldsymbol{I}_{9\times9}$ | $10^{-4}\boldsymbol{I}_{6\times6}$ | $10^{-1}$ | $10^{-4}$ |
| MPTTC | $10^7\boldsymbol{I}_{9\times9}$ | $1.5\cdot10^5\boldsymbol{I}_{9\times9}$ | $10^{-4}\boldsymbol{I}_{6\times6}$ | | |

## C. *Spiral Results*

In Fig.3 we see end-effector position of the MPFC following the described downward spiral. In Fig.4 we see the same for the MPTTC. Both controllers start a small distance from the start of the path. In Fig.5 we see the norms of $\boldsymbol{e_p}$ for the MPFC.In Fig.6 we see the norms of $\boldsymbol{e_p}$ for the MPTTC. Note that the MPFC converges faster than the MPTTC stemming from the MPFC first handling orientation and position before moving along the path. This is a trait of the path-following dynamics and can be adjusted by tuning $q_s$ and $r_v$. The MPTTC on the other hand only has the positions of the desired points at each timestep, and will struggle to catch up with the desired orientation while also moving along the path.

It was observed that the run time of the solver depended on the configuration of the robot and deviation from the path. This is likely due to the solver entering local minima when solving. In Fig.7 we see the run times of the MPFC solver over time during the spiral path test. The run time is slightly longer before it has reached the path. When the path is reached, the

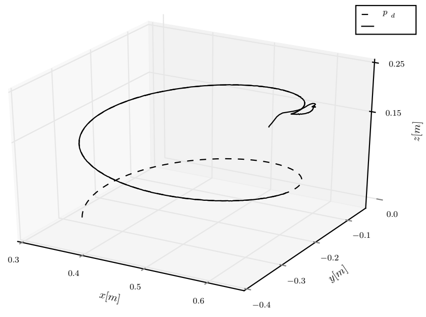


Fig. 3: The MPFC moves to the path before moving along it. The rotation to fit to the paths initial reference frame moves the origin a little off from the path until it converges.

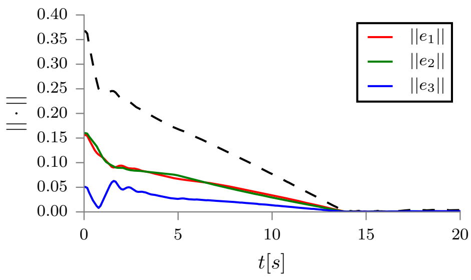


Fig. 5: Norm of the errors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ for the MPFC. The black stippled line is the sum of the norms.

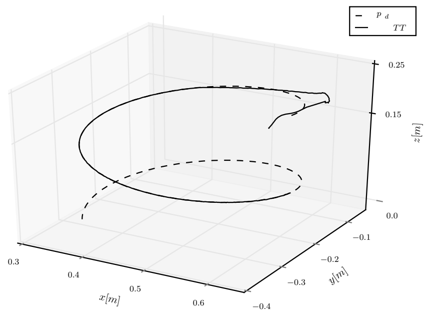


Fig. 4: The MPTTC moves along the path before matching the orientation. The deviation from the path is greater than for the MPFC.

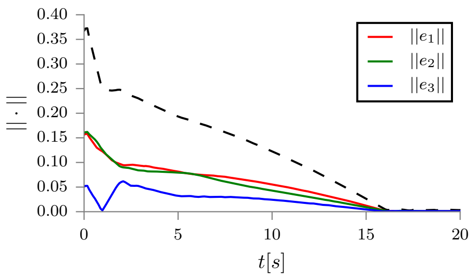


Fig. 6: Norm of the errors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ for the MPTTC. The black stippled line is the sum of the norms.

run time stays relatively consistent. In Fig.8 the run times of the MPTTC is given. We see that the MPTTC solver struggles more than the MPFC when far from the path, but becomes more consistent when on the path.

*Remark* 1. In simulations $\dot{e}_s$ was not required in (11) or (12) for convergence. The delay caused by the solver and the robot interface made the proportional control result in the manipulator oscillating greatly around the path. The introduction of dampening along the path through $\dot{e}_s$ was fundamental for the implementation.

## D. Robot Change

In Fig.10 we see the UR3 and UR5 moving to the 3D Lissajous path

$$\boldsymbol{p}_{bd}^{b}(s) = \begin{bmatrix} 0.035\cos(10s) + 0.1 \\ 0.035\cos(30s+1) + 0.2 \\ 0.035\cos(20s+1) + 0.3 \end{bmatrix} \tag{23}$$

which can be seen in Fig.9. The path was chosen to be as far from the home position $\boldsymbol{q}(0) = [0, -\pi/2, 0, -\pi/2, 0, 0]^T$, and

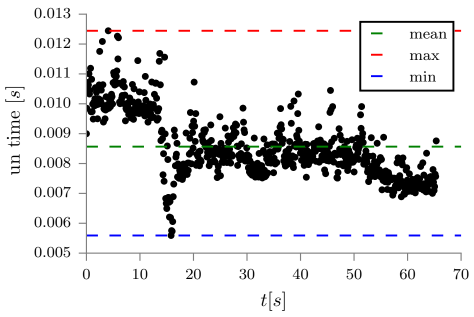


Fig. 7: Run times of the MPFC solver during execution of the path.

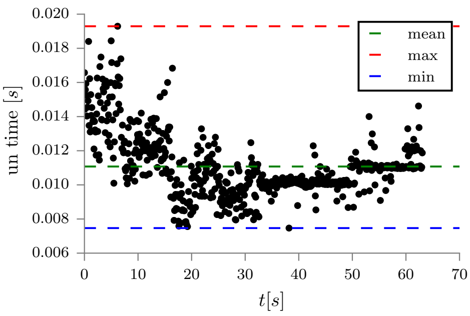


Fig. 8: Run times of the MPTTC solver during execution of the path.

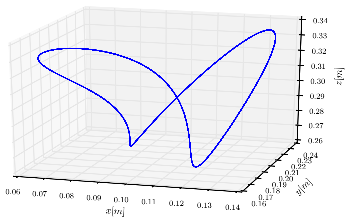


Fig. 9: The 3D Lissajous reference path is placed such that it is inside both the UR3 and the UR5 workspace.

small enough to be within the UR3's workspace. No change in tuning parameters was needed between the two robots. As the previous path, the desired rotation places the end-effector with its $z$-axis pointing downwards.

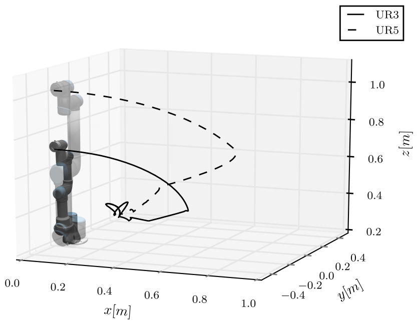


Fig. 10: The UR3 and UR5 start in their upright position, and move to the Lissajous path.

## IV. Discussion and Future work

Both the MPFC and the MPTTC managed to follow the 6 DOF paths. The MPFC first moves to the initial path orientation before moving along the path. The MPTTC on the other hand has a moving setpoint that it must catch up with. The alternative to this is to have one controller to move the end-effector to the desired path and then switching to the along path controller. Using the MPFC allows us to tune the transition from approach to along path motion through the parameter $q_s$.

For a system where switching between active constraints does not happen often and rapidly, the interior point solver may not be the best option. Other solvers which benefit more from warm-starting may prove to give better run times. We believe the times when the solver takes longer than usual is a result of the restoration from local minima. We suggest testing with other solvers for evaluating the useability further. The longer run time of MPTTC stems from how it is implemented, the timing parameters $s$ is included as a parameter, and we believe this may increase the run time slightly.

Delay in the interface and from the solver caused issues for the implementation of the system. The robot would only change joint velocity when a new command was sent, and as the solver could at times run longer than expected, there would be an integration error. This would cause the system to repeatedly overshoot, and oscillate around the desired position. With a faster implementation this is expected to be manageable, but dampening may still be desired.

## V. Conclusion

Using MPC controllers as a mid-level controller between the path generator and the low-level controller allows us the flexibility of changing robots for the same task. Path generation is not necessarily the same as obstacle avoidance, and relinquishing that control to a system between the fast joint controller and the path generator may make for more flexible systems.

The definition of a 6 DOF path as three orthogonal points moving in space was useful for making the MPFC and MPTTC. We also experimentally demonstrated that the MPFC and the MPTTC were capable of following the desired paths.

## VI. Acknowledgements

The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

## References

[1] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, no. 3, pp. 8642–8647, 2009.

[2] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[3] F. Debrouwere, W. Van Loock, G. Pipeleers, M. Diehl, J. Swevers, and J. De Schutter, “Convex time-optimal robot path following with Cartesian acceleration and inertial force and torque constraints,” *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 227, no. 10, pp. 724–732, nov 2013.

[4] T. Faulwasser and R. Findeisen, “Nonlinear Model Predictive Control for Constrained Output Path Following,” *IEEE Transactions on Automatic Control*, vol. 9286, no. c, pp. 1–1, 2016.

[5] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, “Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot,” *IEEE Transactions on Control Systems Technology*, pp. 1–7, 2016.

[6] B. Houska, H. J. Ferreau, and M. Diehl, “ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization,” *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[7] D. Lam, C. Manzie, and M. Good, “Application of model predictive contouring control to an X-Y table,” *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 18, pp. 10 325–10 330, 2011.

[8] M. Böck and A. Kugi, “Real-time nonlinear model predictive path-following control of a laboratory tower crane,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2014.

[9] M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, “On Model Predictive Path Following and Trajectory Tracking for Industrial Robots,” in *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017.

[10] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Trondheim: Marine Cybernetics, 2003.

[11] A. Wächter and L. T. Biegler, *On the Implementation of Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming*, 2006, vol. 106, no. 1.

[12] J. Andersson, “A General-Purpose Software Framework for Dynamic Optimization,” PhD thesis, Arenberg Doctoral School, KU Leuven, Belgium, 2013.

[13] “Actual center of mass for robot - 17264 | Universal Robots.” [Online]. Available: https://www.universal-robots.com/how-tos-and-faqs/faq/ur-faq/actual-center-of-mass-for-robot-17264/

# Paper 4 A System Architecture for Constraint-Based Robotic Assembly with CAD Information

M. H. Arbo, Y. P. Pane, E. Aertbeliën, and Wilm Decré, “A System Architecture for Constraint-Based Robotic Assembly with CAD Information,”*IEEE International Conference on Automation Science and Engineering (CASE)*, Munich, 2018, pp. 690-696.

# A System Architecture for Constraint-Based Robotic Assembly with CAD Information

Mathias Hauan Arbo*, Yudha Pane†, Erwin Aertbeliën† and Wilm Decré†
*Department of Engineering Cybernetics
NTNU, Norwegian University of Science and Technology
†Robotics Research Group, core lab Flanders Make
Department of Mechanical Engineering, KU Leuven

***Abstract*—A system architecture is presented to generate sensor-controlled robot tasks from knowledge encoded in a CAD model. This architecture consists of an application layer where the user annotates assembly tasks in the CAD software. A process layer infers the specific robot skills and parameters from the CAD model and annotated data. A control layer executes the complex, force-controlled tasks. A proof-of-concept implementation is made, consisting of an application layer implemented in FreeCAD and a process layer that focuses on using fuzzy inference to generate appropriate skill-dependent process parameters from the geometric CAD information and annotations in the CAD model. In the control layer, a constraint-based control framework is used to robustly execute the assembly tasks. The system is validated on a challenging assembly task involving the assembly of screw compressor parts.**

## I. Introduction

Manufacturing automation is entering a new era, where a high degree of customization of the product is expected by the client to which manufacturers must adapt quickly. This era is associated with human-robot collaborative workcells, short time to deploy, and tighter coupling of design and manufacturing.

Computer-Aided Design (CAD) software is a rich source of information for robotic assembly processes that may benefit a broad range of companies. The information includes part dimensions, geometric features, contact situations, etc. An early constraint-based description of pose relations between geometric features on assembly parts was presented by Ambler and Popplestone [1]. Reference frames attached to the geometric features were used to define the end-product of assembly in terms of “fits” or “against” relationships. The relative pose between these frames was described by equality or inequality constraints.

Autopass [2] is an early CAD-based assembly programming system that starts from a workpiece rather than a robot-centric perspective. Archimedes 2 [3] describes an architecture where the CAD model and assembly description, sequence planning, and skill execution are separated into modules. One of the most advanced systems is HighLap [4] where the CAD model is annotated with a small set of simple semantic assembly descriptions. The descriptions give constraints similar to Ambler and Popplestone that are used to find remaining degrees-of-freedom. HighLap uses the task frame formalism to execute force-controlled motion [5]. Neto et al. argued that integration of robot path programming in CAD software would benefit small and medium-sized enterprises where programming costs hinder automation [6]. A more recent approach by Perzylo et al. uses object-centric programming based on geometric constraints between parts to simplify the robot motion programming [7]. The comparison of the object-centric programming and classical teach-pendant based programming shows that object-centric programming is faster.

The robot control layer of a CAD-based assembly system can simplify or complicate the architecture. We advocate for: a workpiece perspective, composability, and sensor integration. A workpiece perspective allows us to specify control relative to the parts. Composability allows us to combine and include aspects of the environment, workpiece, and robot. To support a larger class of assembly situations, force control and easy sensor integration are key. Mason’s task frame formalism [8] presents an early force and position control in the task domain. It shows how different control modes can be applied independently along an instantaneous task frame’s directions, both in translation and rotation.

Motions are easily defined w.r.t. the workpiece, environment or robot using constraints, and the constraints are naturally composable. In [9], De Schutter et al. describe a procedure to design a robot controller that can deal with sensor interactions and motion in contact, using a set of auxiliary frames to systematically describe the constraints. This approach gave rise to the iTaSC software framework [10], a systematic approach for constraint-based programming that allows us to specify complex sensor-based skills. Subsequently, expression graphs are used to simplify constraint-based programming leading to a specification language eTaSL [11]. eTaSL scripts are used to specify the continuous behavior of the controller using constraints that relate to geometry or sensor-input. eTaSL scripts can specify monitors that trigger events into a restricted finite state machine (rFSM) [12]. This rFSM describes the discrete switching between different continuous control actions. A control layer using eTaSL/eTC and rFSM can perform assembly tasks in dynamic and uncertain environments, but requires specific process parameters to be defined, such as the force magnitude during assembly or the dither amplitude during insertion. It is difficult to completely determine appropriate values for these parameters in a model-based way, so they are typically tuned. In this paper we want to facilitate the generation of assembly skills from CAD data by automatically determining process parameters using a fuzzy inference module.

The key contribution of this paper is the design of an architecture and a prototype implementation that allows us to generate controllers for assembly tasks requiring complex sensor-based interaction. This is done by splitting the required parameters for these assembly skills into two groups. One group of application and geometry-related parameters is generated from CAD model information. Another group of process parameters is determined using a fuzzy inference module. The resulting parameters are then used to generate a skill specification. These specification files can then be used to execute the specified skill on a reactive constraint-based robot controller. Using this architecture, complex force-controlled assembly skills can be executed even though the skills rely on empirical parameters.

Section II describes the system architecture and its basic concepts such as skills and tasks, and the inference module. Section III describes the software and hardware of the experimental setup, CAD workbench, the implemented skills and parameter inference. The experimental results are presented in Section IV. Section V discusses the experimental results.

## II. System Architecture

The system architecture is outlined in Fig.1 and is split up into three layers: the application layer, the process layer, and the control layer. In the *application layer* we have a workpiece-centric view, where the user annotates the CAD model with the assembly tasks. In the *process layer*, assembly is considered from the point of view of the robotcell, and a planning and inference system ensures appropriate selection and composition of robot skills. The skills involved have application parameters and process parameters. Application parameters can be extracted from the CAD data, e.g. feature frame, or insertion length. Process parameters relate to the CAD data but without a clear underlying model. They may be empirical or have a range of values that produce acceptable results e.g. insertion force, or amplitude of anti-jamming dither. The inference module generates the value of these parameters. In the *control layer* the appropriate eTaSL skills are loaded together with the application parameters and the process parameters. A finite-state machine handles execution of the discrete states.

### A. Skill and Task

Skill and task are often used interchangeably in the robotics literature. In this paper the terms refer to two different concepts.

A *task* is a piece of work to be undertaken, a *skill* is a particular ability. Assembly tasks are high-level assembly specifications in the application layer, and skills are related to particular actions the robot can perform. We differentiate between atomic skills and composed skills. The skills are defined by a set of specification files.

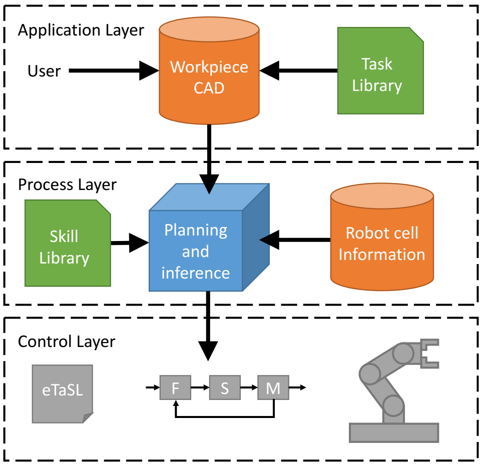


Fig. 1. System architecture.

*Atomic skills* are eTaSL scripts that have: a configuration, inputs, outputs, and event specifications. The configuration is a set of parameters that are prepended to the eTaSL script and are constant during skill execution. These could be application parameters coming from the CAD model or process parameters generated by the inference module. The inputs bring information from continuous sources such as sensors. The outputs are mainly used for logging or analysis purposes. Monitors can trigger events that denote success or failure modes of the skill.

A common approach in robotics is to have states in finite state machines denote motion goals, e.g. *move_to_grasp_location*. This puts the skill complexity in terms of the states. An approach more often found in computer games is to have states denote operational modes, e.g. *walk*, *run*, or *move_cartesian*, and then redefine the parameters of these states during runtime. This reduces the states used and puts the complexity in the transitions. We have used the operational modes approach as it is easy to reuse a finite state machine with new application parameters, and as transition events denote the momentary situation that the robot is in. A *composed skill* is therefore an rFSM where each state corresponds to an atomic skill to be executed, and a function call associated with each transition event. The function call invokes external services such as grippers and tools, and loads the configuration parameters of the next atomic skill to be executed.

From a workpiece perspective, inserting a peg into a hole with large or small clearance is the same task, but they may need completely different insertion control strategies, e.g. when there is a large clearance a pure position-based approach could be sufficient, while small clearance would necessitate force-control strategies. Therefore we map a task to a set of potentially applicable composed skills.

Going from task to composed skill is done by first finding the geometric primitives involved, and then using the CAD data describing these primitives to select the composed skill from a set of composed skills associated with a task on such geometric primitives, see Fig.2. In conclusion, tasks are generalisable and workpiece-centric, while skills correspond to an ability of the robot system to realize a task in a specific way.

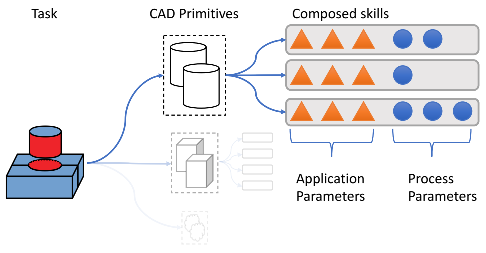


Fig. 2. A task is annotated between two geometric primitives. This task for these geometric primitive types can be performed by a set of composed skills. The CAD data of the geometric primitives is used to select the composed skill, which has application parameters and process parameters.

### B. Planning and Inference Module

The planning and inference module has three main purposes: plan the assembly sequence, compose skills based on the task and the workcell, and generate the appropriate configuration parameters. Optimal planning of assembly involves optimal task sequencing, optimal trajectory planning, and selecting the optimal parameter values. These domains are interconnected, but they are assumed separable in the scope of this article. This article does not give a planning strategy but outlines how the task annotations tie in to the literature on assembly sequences.

*1) Assembly Sequence Planning:* tasks and parts form a *liaison graph*. In liaison graphs, a node represents a part, and an edge represents an assembly situation [13]. A *precedence graph* specifies which edges in the liaison graph should be completed before others. This precedence graph represents all feasible assembly sequences. In [14], Homem De Mello and Sanderson show the relation between precedence graphs and other assembly representations such as And/Or graphs. The precedence graph is created in the application layer. Assembly of geometric features gives a defined assembly direction and virtual disassembly in the CAD software along these lines provides a suggested precedence graph. The precedence graph is passed from the application layer to the planner for generation of the assembly sequence. Moving to grasp, changing tool, reorientation, and other workcell-related composed skills are added to the assembly sequence by the planner. Robot cell information such as which parts and tools are available, or are collaborative workspace skills required, is an essential plan of the planning module. This information must be stored in a robot cell database describing the different setups available for the manufacturer. As the underlying control layer is robot-agnostic and the skills are transferrable, we view this as a problem to be addressed in the planning module.

*2) The Parameter Inference Module:* given the geometric information between two mating parts and the selected skill, the inference module generates the appropriate process parameters to ensure successful assembly. Here we present an inference module for the insertion task of a cylindrical object in a tight-tolerance situation. This task would be extremely difficult to perform with only position control, hence the necessity of a force-controlled skill. We generalize the task as a peg-in-hole problem.

A number of process parameters need to be tuned for optimal insertion behavior. From existing literature [15], it is known that the insertion behavior is determined by a number of factors such as the peg's dimension and the peg-hole clearance. Even for a peg-in-hole assembly with relatively simple geometry, accurate process modeling is difficult to achieve for narrow clearances. Due to hyperstatic contact situations, not all contact forces are always externally observable. We therefore use a data-driven approach to determine the process parameters. To infer parameters in an uncertain or stochastic situation, a fuzzy inference approach is chosen [16]. Our fuzzy inference module uses the peg's length, peg's diameter and clearance to estimate the appropriate insertion force and dither amplitude.

## III. Implementation

### A. Assembly Use Case

We consider an assembly of a large and a small rotary-screw compressor (see Fig.3). The large compressor is composed of >30 parts and the small compressor is composed of >15 parts. To limit the scope of this demonstration, we focus on the assembly of the rotors and the housing lid. The housing is attached to a fixture and all insertions are in the same direction. Each compressor has a small and large meshing helical screw rotors. The top of the rotors go through the housing lid trough two chamfered holes. All parts to be assembled are placed in known poses. Fig.8 shows the resulting assembly sequence.

### B. Application Layer

The application layer is implemented as a workbench in FreeCAD 0.16 [17]. FreeCAD is an open-source parametric CAD program. We implemented three task classes: *Insert*, *Place*, and *Screw*. Each of them is associated with two faces, and two *FeatureFrame* objects. To transfer information about the geometric features and their location, we implemented a tool for creating reference frames on geometric primitives. *FeatureFrame* objects can be instantiated on a selected vertex, edge, or face. If the selected geometric primitive has a center, center of mass, axis, or focus, the feature frame can be placed at the attribute with $z$-axis oriented along the axis if possible. On edges the frame can be placed along the edge

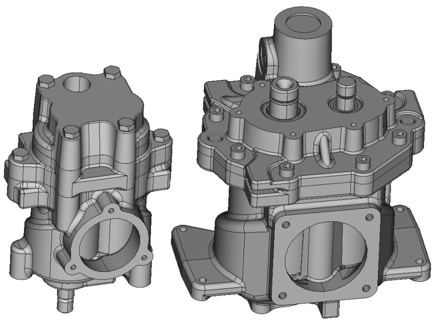


Fig. 3. CAD model of the two compressors side by side.

with $x$-axis aligned with the edge tangent. On the face the frame is placed with $z$-axis aligned with the normal. The parts are exported as STL meshes for visualization, and a JSON file describing the feature frames and the attributes of the geometric primitives relative to the mesh origin. The feature frame part of the workbench is publicly available [18]. By creating task instances between parts we form a liaison diagram, see Fig.4. During instantiation of an *Insert* or *Screw* object, we also instantiate a *FeatureFrame* object denoting the instantaneous task frame for force control. Since assembly sequence planning is not the focus of this paper, the assembly sequence was manually determined while specifying the task. The task instances, in order of execution, are: *insert_littlerotor*, *insert_bigrotor*, and *place_lid*. The tasks are exported in JSON files with reference to the part names and feature frame names. Grasp location is assumed to be known and is annotated as a *FeatureFrame* instance on the part.

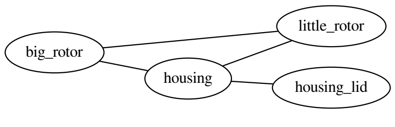


Fig. 4. Liaison diagram for the assembly use case. Although the rotors are in contact with the lid, the chamfers allow us to use a place task defined between the housing and the lid.

### C. Implemented Skills

In this use-case each task maps to a single composed skill. When the skill set of the system grows this will no longer be the case. The two implemented composed skills are composed of a larger number of atomic skills, as listed in Table I. As the table shows, the *insert_littlerotor* instance uses *grasp_and_insert* as the suitable composed skill. This composed skill consists of three different atomic skills: *move_cartesian*, *guarded_cartesian*, and *cylinder_insert*. The transition between these atomic skills is shown in Fig. 5. When *e_start* occurs, we set the goal of *move_cartesian* to reach a position offset a constant distance along the $z$-axis of the grasp frame. Success in reaching the goal is associated with the *e_pregrasp* event. When *e_pregrasp* occurs the goal of *move_cartesian* is set to the grasp frame with success of the atomic skill being associated with *e_grasp*. When *e_grasp* occurs we have reached the grasp pose and engage the gripper and set the next goal to a $z$-axis offset from the grasp frame that we call post-grasp, and success gives the *e_postgrasp* event. The procedure is repeated for the $z$-axis pre-hole locations. When *e_prehole* occurs we transition into *guarded_cartesian* which is a cartesian motion towards the hole with monitors that trigger when end-effector force exceeds a threshold. This procedure of reconfiguring the goals of the atomic skill is adhered to for all of the transitions.

TABLE I
PROTOTYPE DESCRIPTION

| Part name | Tasks | Composed Skill | Atomic Skills | Tool |
|---|---|---|---|---|
| *small_rotor* | *insert* | *grasp_and_insert* | *move_cartesian* *guarded_cartesian* *cylinder_insert* | *gripper* |
| *big_rotor* | *insert* | *grasp_and_insert* | *move_cartesian* *guarded_cartesian* *cylinder_insert* | *gripper* |
| *housing_lid* | *place* | *grasp_and_place* | *move_cartesian* *guarded_cartesian* | *gripper* |

As described in Section II, each atomic skill is implemented as an eTaSL script. This script contains a number of constraints, monitors and input/output ports. For example, in the *cylinder_insert* skill, a constraint is used to impose a specified insertion force. The constraint is parameterized by the instantaneous task frame and desired insertion force $F_{z,des}$ along the cylindrical axis. Meanwhile zero forces and torques are maintained along and around the other axes. The location of the instantaneous task frame is a *FeatureFrame* instance created on the peg-face of the *Insert* instance, this is an an application parameter of the composed skill. The target insertion force is a process parameter given by the inference module. The dither is applied as superposed torques around the $x$ and $y$ axes of the instantaneous task frame. The selection of the dither axes is based on empirical results.

### D. Inference Module

A Mamdani-type fuzzy inference is chosen and implemented in MATLAB R2017a's fuzzy logic designer toolbox. 21 rules are defined, describing the mapping from the peg's dimension and the clearance to the appropriate insertion force and dither amplitude. The fuzzy rules are derived from rough modeling of the contact situation and the empirical trend observed after executing the *cylinder_insert* skill. Each fuzzy rule is approximated with gaussian membership functions

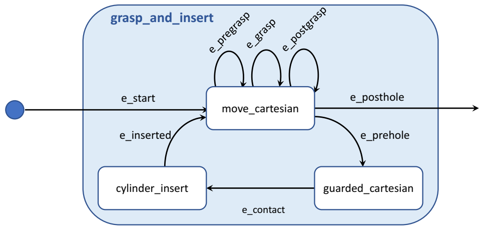


Fig. 5. Example of the finite state machine for *grasp_and_insert*. The transition between two states is triggered by incoming event "e_event_name".

that categorize a range of parameter values into fuzzy sets. For example, the peg's diameter can be categorized into "*very small*", "*small*", "*medium*", "*large*" and "*very large*". The fuzzy sets for each geometric and process parameters are shown in Fig. 6 while five sample rules are provided in Listing 1.

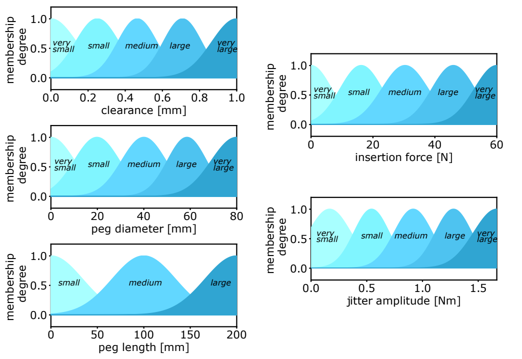


Fig. 6. The fuzzy sets used in the inference module. The left column shows the geometric parameters while the right one shows the process parameters.

Listing 1. Selection of the fuzzy rules used in the inference module

```
if diameter is "small" then insertion force is "large"
if clearance is "medium" then insertion force is "medium"
if length is "long" then insertion force is "small"
if clearance is "very small" then dither amplitude is "very large"
if diameter is "small" and length is "long" then dither amplitude is "medium"
```

In this data-driven approach, we first collected the training data by running a total of 40 peg-in-hole trials with different process parameter values. For each run, the insertion force and dither amplitude were carefully selected to ensure successful execution, while minimizing contact force. Based on these experiments, the number of membership functions and their shapes were then tuned. Finally, the fuzzy inference module was validated with a new set of peg-in-hole tasks.

### E. Control Layer

The experimental setup consists of a seven DOF KUKA LBR iiwa 14 robot equipped with a pneumatic SCHUNK RH940 parallel gripper. The robot is controlled with eTaSL/eTC [11] which is deployed as an Orocos [19] component. eTaSL is used for constraint specification in Lua language while eTC is the controller implementation that satisfies the constraints in an instantaneously optimal way. The control loop is run with a frequency of 200 Hz. Additionally, a *feature_frame_publisher* node [20] was created in ROS [21] for publishing all relevant frames with TF2. Actuation of the gripper is also performed through a ROS service call.

## IV. Experimental Results

The overall system is deployed for assembling the small rotor, big rotor, and housing lid consecutively. The rotor assemblies are categorized as tight-tolerance tasks, therefore the *grasp_and_insert* composed skill using force control is selected, and process parameter inference is required. The geometric property generated from the CAD information of the tasks and their inferred process parameters are reported in Table II. Meanwhile, since the housing lid assembly has a relatively large clearance, a position control skill with force monitor i.e. *guarded_cartesian* is sufficient. Therefore, process parameter inference is not required.

The experiment shows that the robot successfully assembles the parts using a parametrized skill by the application and process layers, see Fig.8 for a snapshot of the assembly sequence. The recorded insertion forces and moments during a portion of the assembly sequence are shown in Fig 7. The plot gives a recording of the transition from *guarded_cartesian* to *cylinder_insert*. To emphasize the behavior of the dithering in the skill, we do not transition from *cylinder_insert* to *move_cartesian*, but remain in the *cylinder_insert* skill.

TABLE II
Geometric and Inferred Process Parameters for the Rotor Assemblies

| Parameter Category | Parameter Name | Part Name | |
|---|---|---|---|
| | | *small_rotor* | *big_rotor* |
| Geometric | Length (mm) | 192 | 192 |
| | Diameter (mm) | 61 | 75.5 |
| | Clearance (mm) | 0.1 | 0.08 |
| Process | Insertion Force (N) | -31.1 | -33.5 |
| | dither Amplitude (Nm) | 1.13 | 1.33 |

## V. Discussion

The force controller in eTaSL has a damping behavior [11]. This resulted in a velocity proportional to the difference between the measured and desired force. The plot in Fig. 7 shows that when the robot approached the rotor's hole between 0 s and 2.8s the forces were approximately zero. When a force magnitude of -10 N in $F_z$ was experienced by *guarded_cartesian* a transition into *cylinder_insert* occurred. The insertion process lasted from 2.8 s to 6.7 s and the robot was moving down with a constant velocity. The effects of friction (approximately -10 N in $z$-direction) are visible

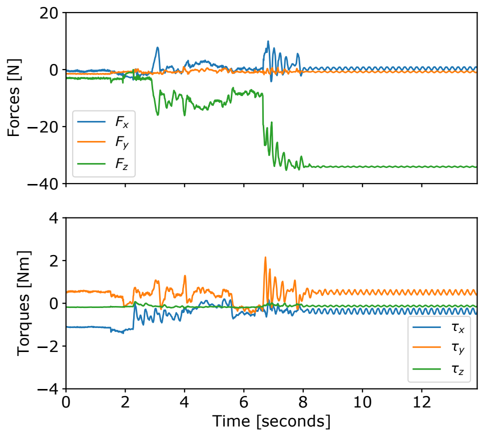


Fig. 7. The measured forces and torques during insertion of the big rotor. Note that from 8 s, the rotor is in contact with the bottom. This was added to verify that the desired dithering is present.

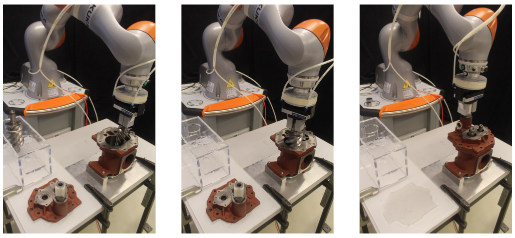


Fig. 8. The robot performing the assembly sequence of the compressor parts. From left to right: assembly of the small rotor, the big rotor, and the housing lid.

during the insertion. The remaining difference between insertion force set-point and the measured friction caused an approximately constant insertion velocity. The dithering was not easily distinguished in the graph during the insertion process. From 6.7 s to 8 s, the rotor touched the bottom of the hole, the velocity dropped to zero, and the set-point for the insertion force reached its desired value. The measured force along the $z$-axis was consistent with the selected process parameters listed in Tab. II. Furthermore, the oscillations due to the applied dither-force remained bounded. This shows that the skill executed the assembly task as desired.

The composed skills are given the names of feature frames. These are used to look up the location of parts using the published frames, making them less dependent on hard-coded locations. An example of this is the grasp frame available on each of the parts. The pre-grasp and post-grasp locations were constant offsets from the grasp frame, and the grasp frame was queried from TF2 based on the part name. This allows easy integration with robot cell localization systems that publish part locations using TF2.

The published frames were also beneficial in calibration of the robot cell as we could use the known CAD data on where the holes are on the parts to quickly calibrate the part location, e.g. we moved the end-effector to the grasp location of a rotor and read where the mesh was located relative to the grasp location from the *feature_frame_publisher*. This can be useful if a teach-in of the robot cell information is desired.

An aspect of the geometric information that was not incorporated was the part symmetry, the rotors are cylindrical and can be grasped or inserted from any orientation around its $z$-axis. This information is available in the JSON file of the part, and requires developing an atomic skill, *move_cartesian_symmetric*, where the symmetric information is added to the application parameters of the skill. This would allow more efficient execution of the approach motions. Depending on the specific mounting of the gripper on the robot, this can also increase the working range of the robot.

In this paper we have emphasized the geometric information available in the CAD data, but there is another aspect that is of relevance, the material properties of the parts involved. For example, insertion of a teflon peg in a steel hole has different process parameters to inserting a steel peg in a soft-plastic hole. Material properties are available in STEP AP214 file format, and many CAD programs support it. For multi-material assembly scenarios, the inference engine should be trained on the material properties as well as the geometric properties.

## VI. Conclusion

In this paper, a three-layered system architecture for automating robot assembly programming using CAD information is proposed. The application layer is used for annotating the tasks, the process layer is used for planning and inference of the relevant robot skills, and the control layer is used for execution of the skills on the robot platform. We focus on more challenging assembly tasks that require force-control. To accomplish this we use skills that have process parameters such as insertion force and dither amplitude. These empirical parameters are not readily available in the CAD model. A fuzzy logic inference module is used to capture this process knowledge by providing a method of inferring the process parameters based on a set of key geometric parameters from the CAD model.

As a proof-of-concept, we implemented the proposed system architecture and validated it with force-controlled insertion of compressor rotors. The problem can be generalized as a peg-in-hole task, for which the conventional solutions have been studied well [22]. The key geometric parameters used in the fuzzy logic was the peg's clearance, peg's diameter, and insertion length, all of which were extracted from the CAD model and Task object in the application layer. For more complex assembly tasks, such as click-connection, the key geometric parameters are yet to be defined.

Inference is realized using a fuzzy inference method which proved to be straightforward to implement. Of course, this method fails to extrapolate outside the predefined range of the fuzzy sets and we assumed that all of the given geometric parameters are relevant. With a sufficiently large dataset and more advanced regression methods (such as automatic relevance detection [23]), it is possible to deduce which of the parameters are really relevant for the process parameters. Such an approach will be beneficial for including material properties when inferring process parameters. The fuzzy set database can grow over time to accommodate a larger variety of assembly cases and this database can be shared with similar robot setups.

For many industrial assembly cases, there are more than two contact surfaces involved. An example is the housing lid and the two rotors. There are two main approaches to this: defining a primitive boundary representation [4], or using the subshape. In this article we have assumed the chosen geometric primitive to be sufficient for completing the task. To handle more complex geometries, a task could be split into subtasks with their individual primitive boundary representations, or a special purpose composed skill can be defined.

The preliminary implementation contains prototypes and tools that the eventual system will use. The assembly of the rotors and housing lid was successfully executed by the system, and the inference module provided reasonable parameters. The application layer implementation is useful for annotating geometric features to CAD models and the publisher node gives their transformations in ROS.

A wealth of research exists on assembly analysis, planning, and skills. This research can become tools that can benefit researchers as well as manufacturers. We describe a system architecture based on previous work that also allows for process parameter inference for tight-tolerance assembly situations and share some of the tools built in the process. However, it is difficult to build up a comprehensive system. Therefore we advocate prototyping in open-source frameworks such as FreeCAD and Orocos.

## VII. Acknowledgement

The work reported in this paper was supported by Flanders Make ICON FINROP (Fast and Intuitive Robot Programming) in Belgium and the centre for research based innovation SFI Manufacturing in Norway. The work is partially funded by the Research Council of Norway under contract number 237900.

## References

[1] A. Ambler and R. Popplestone, "Inferring the positions of bodies from specified spatial relationships," *Artificial Intelligence*, vol. 6, no. 2, pp. 157–174, jun 1975.

[2] L. I. Lieberman and M. A. Wesley, "AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly," *IBM Journal of Research and Development*, vol. 21, no. 4, pp. 321–333, jul 1977.

[3] S. Kaufman, R. Wilson, R. Jones, T. Calton, and A. Ames, "The Archimedes 2 mechanical assembly planning system," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4. IEEE, 1996, pp. 3361–3368.

[4] U. Thomas and F. M. Wahl, "Assembly Planning and Task Planning — Two Prerequisites for Automated Robot Programming," in *Springer Tracts in Advanced Robotics*, 2010, vol. 67, pp. 333–354.

[5] F. Dietrich, J. Maaß, A. Raatz, and J. Hesselbach, "RCA562: Control Architecture for Parallel Kinematic Robots," in *Springer Tracts in Advanced Robotics*, 2010, vol. 67, pp. 315–331.

[6] P. Neto, N. Mendes, R. Araújo, J. Norberto Pires, and A. Paulo Moreira, "High-level robot programming based on CAD: dealing with unpredictable environments," *Industrial Robot: An International Journal*, vol. 39, no. 3, pp. 294–303, apr 2012.

[7] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-Novem. IEEE, oct 2016, pp. 2293–2300.

[8] M. T. Mason, "Compliance and Force Control for Computer Controlled Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, 1981.

[9] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, may 2007.

[10] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "iTASC: a tool for multi-sensor integration in robot manipulation," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, aug 2008, pp. 426–433.

[11] E. Aertbelien and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2014, pp. 1540–1546.

[12] M. Klotzbücher and H. Bruyninckx, "Coordinating robotic tasks and systems with rfsm statecharts," *JOSER: Journal of Software Engineering for Robotics*, vol. 3, pp. 28–56, 2010.

[13] T. De Fazio and D. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 640–658, dec 1987.

[14] L. Homem de Mello and A. Sanderson, "Representations of mechanical assembly sequences," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 211–227, apr 1991.

[15] N. Pitchandi, S. P. Subramanian, and M. Irulappan, "Insertion force analysis of compliantly supported peg-in-hole assembly," *Assembly Automation*, vol. 37, no. 3, pp. 285–295, 2017. [Online]. Available: http://www.emeraldinsight.com/doi/10.1108/AA-12-2016-167

[16] S. Guillaume, "Designing fuzzy inference systems from data: An interpretability-oriented review," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 426–443, jun 2001.

[17] J. Riegel and Y. van Havre, "FreeCAD: Parametric 3D modeler." [Online]. Available: https://www.freecadweb.org

[18] M. H. Arbo and Y. Pane, "ARBench," 2017. [Online]. Available: https://github.com/mahaarbo/ARBench

[19] P. Soetens, "A Software Framework for Real-Time and Distributed Robot and Machine Control," Ph.D. dissertation, Katholieke Universiteit Leuven, 2006.

[20] M. H. Arbo and Y. Pane, "ARBench part publisher," 2017. [Online]. Available: https://github.com/mahaarbo/arbench{_}part{_}publisher

[21] M. Morgan Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, may 2009.

[22] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

# Paper 5 Stability of the Tracking Problem with Task-Priority Inverse Kinematics

M. H. Arbo and J. T. Gravdahl, “Stability of the Tracking Problem with Task-Priority Inverse Kinematics”, *IFAC Symposium on Robotics and Control (SY-ROCO)*, Budapest, 2018, pp. 121-125.

# Stability of the Tracking Problem with Task-Priority Inverse Kinematics [★]

**Mathias Hauan Arbo** [*] **Jan Tommy Gravdahl** [**]

[*] *Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway (e-mail: mathias.arbo@ntnu.no).*

[**] *Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway (e-mail: jan.tommy.gravdahl@ntnu.no).*

**Abstract:** The multiple task-priority inverse kinematics framework using the Moore-Penrose pseudoinverse has been shown to be asymptotically stable for the regulation problem with certain conditions on the tasks. In this article we present a theorem that extends this to the tracking problem by including an additional criterion that we term *fully represented in the null-space*. We show the effect of this on a simulation with a snake-like robot manipulator with 30 links for 3 compatible tasks, and an example of 2 tasks that are compatible as a regulation problem but incompatible as a tracking problem. As the tracking problem is more affected by the linearization assumption, we also include an example showing that the effect of linearization can be detrimental during tracking.

*Keywords:* Inverse kinematic problem, Stability analysis, Control (closed-loop), Robot kinematics, Industrial robots, Robot arm

## 1. INTRODUCTION

From humanoid service robots to industrial manipulators, we often want the robot to follow a reference trajectory. The reference is represented in a task space (f.ex. in Cartesian space) and finding the appropriate robot-centric control setpoints to converge to the reference is referred to as the inverse kinematics problem. Inverse kinematics is a classical problem in robotics, and is fundamental to highly redundant robots such as humanoid or snake robots.

Sciavicco and Siciliano (1986) and Das et al. (1988) present methods of finding joint speed setpoints to achieve a given inverse kinematics. This is commonly called the closed-loop inverse kinematics (CLIK), where the joint speeds are found by inverting differential kinematics. By design, the joint speeds are chosen such that the task errors converge exponentially.

For robots where there are multiple tasks to be achieved, and the robot is redundant with respect to the tasks, Chiaverini (1997) shows that tasks can be combined in priority by placing lower priority tasks in the null-space of the higher priority tasks. Antonelli (2009) presents a set of criteria on the tasks to ensure that we have asymptotic stability of the regulation problem (when the robot is to move to a reference point).

Multiple-task inverse kinematics can be solved in many different ways. We consider the use of pseudo-inverses and null-space operators, but others such as Aertbeliën and Schutter (2014) solves the problem by using a quadratic program. Falco and Natale (2011) consider the discretized version of the problem, and gives a stability proof of the regulation problem.

In this article we consider the tracking problem where the robot is to follow a reference trajectory rather than the regulation problem where it moves to a point. We present a proof with criteria on the tasks to ensure that the tracking problem exhibits asymptotic stability. We give three examples on a 30 link planar snake robot. The first example emphasizes the behavior when we have compatible tasks. The second is a minimal example of tasks that are compatible as a regulation problem but incompatible as a tracking problem. The minimal example can easily happen if a user makes a mistake when designing the tasks. The final example shows how the tracking problem is more sensitive to how long the joint speed setpoint is applied to the robot than the regulation problem.

## 2. STABILITY

### *2.1 Description of the Moore-Penrose Pseudoinverse*

The Moore-Penrose pseudoinverse, $\boldsymbol{A}^\dagger$ is defined for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ where $\boldsymbol{A}$ not necessarily has full rank, and it satisfies the following conditions:

$$\boldsymbol{A}\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{A}, \tag{1}$$

$$\boldsymbol{A}^\dagger\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{A}^\dagger, \tag{2}$$

$$\left(\boldsymbol{A}\boldsymbol{A}^\dagger\right)^T = \boldsymbol{A}\boldsymbol{A}^\dagger, \tag{3}$$

$$\left(\boldsymbol{A}^\dagger\boldsymbol{A}\right)^T = \boldsymbol{A}^\dagger\boldsymbol{A} \tag{4}$$

[★] The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

and if $\boldsymbol{A}$ has full rank, we have

$$\boldsymbol{A}\boldsymbol{A}^{\dagger} = \boldsymbol{I}_m. \tag{5}$$

## 2.2 Single task stability

We consider a general robotic system with $n$ degrees of freedom whose configuration is described by the joint coordinates $\boldsymbol{q} = [q_1, q_2, \dots, q_n] \in \mathbb{R}^n$. The robot is controlled by the speed of the joints $\dot{\boldsymbol{q}}_{\text{des}}$ and we assume $\dot{\boldsymbol{q}}_{\text{des}}(t) = \dot{\boldsymbol{q}}(t)$. We define a task error, $\tilde{\boldsymbol{\sigma}} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$, that should be stabilized to zero. It is a function of time, joint coordinates, and $m$ is the dimension of the task. A typical regulation problem would be:

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{p} - \boldsymbol{f_p}(\boldsymbol{q}) \tag{6}$$

where $\boldsymbol{f_p}$ is the forward kinematics, and $\boldsymbol{p}$ is the desired goal state. A typical tracking problem would be:

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{p}_{\text{ref}}(t) - \boldsymbol{f_p}(\boldsymbol{q}) \tag{7}$$

where $\boldsymbol{p}_{\text{ref}}(t)$ is a reference signal that the end-effector tracks.

Looking at the total derivative of $\tilde{\boldsymbol{\sigma}}$ we find the task error is related to the control input $\dot{\boldsymbol{q}}$ by

$$\dot{\tilde{\boldsymbol{\sigma}}}(t, \boldsymbol{q}) = \frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial t}(t, \boldsymbol{q}) + \frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial \boldsymbol{q}}(t, \boldsymbol{q})\dot{\boldsymbol{q}}(t) \tag{8}$$

we define $\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial \boldsymbol{q}} \in \mathbb{R}^{m \times n}$ as the configuration-dependent task error Jacobian. For brevity we simply write $\boldsymbol{J}$ and $\tilde{\boldsymbol{\sigma}}$. We essentially assume that we can linearize the system at this time instance and give the robot a $\dot{\boldsymbol{q}}$ setpoint for a short enough time interval that we have not diverged from this linearization.

To ensure exponential stability of the task error, choose

$$\dot{\tilde{\boldsymbol{\sigma}}} := -\Lambda \tilde{\boldsymbol{\sigma}}, \tag{9}$$

where $\Lambda > 0$. From (8), we find

$$\boldsymbol{J}\dot{\boldsymbol{q}} = -\frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial t} - \Lambda \tilde{\boldsymbol{\sigma}}. \tag{10}$$

Note that $\frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial t}$ is a feedforward term of the time derivative of the task error. If $\boldsymbol{J}$ has full rank, we can use the Moore-Penrose inverse of $\boldsymbol{J}$ to find the desired $\dot{\boldsymbol{q}}$

$$\dot{\boldsymbol{q}} = -\boldsymbol{J}^{\dagger}\left(\frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial t} + \Lambda \tilde{\boldsymbol{\sigma}}\right). \tag{11}$$

if the system is redundant with respect to the task, i.e. $n > m$, the solution has a null-space operator $\boldsymbol{N} = (\boldsymbol{I}_n - \boldsymbol{J}^{\dagger}\boldsymbol{J})$ such that

$$\dot{\boldsymbol{q}} = -\boldsymbol{J}^{\dagger}\left(\frac{\partial \tilde{\boldsymbol{\sigma}}}{\partial t} + \Lambda \tilde{\boldsymbol{\sigma}}\right) + \boldsymbol{N}\dot{\boldsymbol{q}}_{\text{null}} \tag{12}$$

where $\dot{\boldsymbol{q}}_{\text{null}} \in \mathbb{R}^n$ can be arbitrarily chosen without affecting the stability of the task. This means that tasks are combined in priority by projecting their desired joint speed, (11), into the null-space of the higher-priority tasks.

We use the augmented null-space operator to achieve this. The augmented null-space operator describes the null-space formed by the combination of multiple task error Jacobians. That is, for task errors $\tilde{\boldsymbol{\sigma}}_1$ and $\tilde{\boldsymbol{\sigma}}_2$ the augmented Jacobian is defined as

$$\boldsymbol{J}_{1,2} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix}, \tag{13}$$

and the augmented null-space operator is defined as

$$\boldsymbol{N}_{1,2} = \left(\boldsymbol{I}_n + \boldsymbol{J}_{1,2}{}^{\dagger}\boldsymbol{J}_{1,2}\right). \tag{14}$$

A useful property of the augmented null-space operator is:

$$\boldsymbol{J}_i \boldsymbol{N}_{1,\dots,k} = \boldsymbol{0}_{m \times n} \tag{15}$$

with $i \in \{1, \dots, k\}$. For proof see Moe et al. (2016). The desired joint speed for $k$ tasks, is then:

$$\dot{\boldsymbol{q}} = -\sum_{i=1}^{k} \boldsymbol{N}_{1,i-1} J_1{}^{\dagger}\left(\frac{\partial \tilde{\boldsymbol{\sigma}}_1}{\partial t} + \Lambda_1 \tilde{\boldsymbol{\sigma}}_1\right) \tag{16}$$

where we have defined the shorthand $\boldsymbol{N}_{1,0} = \boldsymbol{I}_n$ to simplify our sum expression.

## 2.3 Task relation definitions

Antonelli (2009) provides three useful definitions for the relation between tasks. Two tasks with Jacobians $\boldsymbol{J}_i$ and $\boldsymbol{J}_j$ are defined to be *annihilating* if:

$$\boldsymbol{J}_i \boldsymbol{J}_j{}^{\dagger} = \boldsymbol{0}_{m \times m}. \tag{17}$$

They are *annihilating in the null-space* of $\tilde{\boldsymbol{\sigma}}_1, \dots, \tilde{\boldsymbol{\sigma}}_l$ if

$$\boldsymbol{J}_i \boldsymbol{N}_{1,\dots,l} J_j^{\dagger} = \boldsymbol{0}_{m \times m}, \tag{18}$$

and the two tasks are *independent* if they are not annihilating and

$$\rho(\boldsymbol{J}_i^{\dagger}) + \rho(\boldsymbol{J}_j^{\dagger}) = \rho\left(\left[\boldsymbol{J}_i{}^{\dagger}, \boldsymbol{J}_j{}^{\dagger}\right]\right) \tag{19}$$

where $\rho(\cdot)$ denotes the rank.

We include an additional definition, a tracking task is *fully represented in the null-space* $\boldsymbol{N}_j$ if

$$\boldsymbol{J}_i \boldsymbol{N}_j \boldsymbol{J}_i^{\dagger} = \boldsymbol{J}_i \boldsymbol{J}_i^{\dagger} \tag{20}$$

which for full rank of $\boldsymbol{J}_i$ gives the identity matrix. This new criteria ensures that the time-varying aspect to track in a task can be followed in the null-space it will operate in.

## 2.4 Tracking Multiple tasks

*Theorem 1.* Given $k > 1$ tasks, and that

(A) each task has *full rank*,
(B) task $i$ is *independent* of all tasks $1, \dots, i-1$,
(C) any tasks $i$ and $j$ with $i > j > 1$ are *annihilating* in the augmented null-space $\boldsymbol{N}_{1,\dots,j-1}$,
(D) if task $i$ is a tracking task, it is *fully represented* in $\boldsymbol{N}_{1,\dots,i-1}$

then the task errors are asymptotically stable.

**Proof.** We have $k > 1$ tasks. For each of these tasks, we desire exponential behavior of the task error derivative

$$\dot{\tilde{\boldsymbol{\sigma}}}_i = \frac{\partial \tilde{\boldsymbol{\sigma}}_i}{\partial t} + \boldsymbol{J}_i \dot{\boldsymbol{q}} \tag{21}$$

$$\dot{\tilde{\boldsymbol{\sigma}}}_i := -\Lambda \tilde{\boldsymbol{\sigma}}_i \tag{22}$$

where $i = 1, \dots, k$. To investigate the stability of all the tasks, we define

$$\overrightarrow{\boldsymbol{\sigma}} = [\tilde{\boldsymbol{\sigma}}_1^T, \dots, \tilde{\boldsymbol{\sigma}}_k^T]^T \tag{23}$$

$$\frac{\partial \overrightarrow{\boldsymbol{\sigma}}}{\partial t} = [\frac{\partial \tilde{\boldsymbol{\sigma}}_1}{\partial t}^T, \dots, \frac{\partial \tilde{\boldsymbol{\sigma}}_k}{\partial t}^T]^T \tag{24}$$

Then, by inserting (16) into (21) we obtain the system

$$\dot{\vec{\sigma}} = -\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{0}_{m_1\times m_2} & \dots & \boldsymbol{0}_{m_1\times m_k} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \dots & \boldsymbol{0}_{m_2\times m_k} \\ \vdots & & \ddots & \\ \boldsymbol{A}_{k1} & \boldsymbol{A}_{k2} & \dots & \boldsymbol{A}_{kk} \end{bmatrix} \vec{\sigma} + \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{0}_{m_1\times m_2} & \dots & \boldsymbol{0}_{m_1\times m_k} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} & \dots & \boldsymbol{0}_{m_2\times m_k} \\ \vdots & & \ddots & \\ \boldsymbol{B}_{k1} & \boldsymbol{B}_{k2} & \dots & \boldsymbol{B}_{kk} \end{bmatrix} \frac{\partial \vec{\sigma}}{\partial t} \quad (25)$$

where

$$\boldsymbol{A}_{ij} = \boldsymbol{J}_i \boldsymbol{N}_{1,\dots,j-1} \boldsymbol{J}_j^\dagger \Lambda_j \quad (26)$$

$$\boldsymbol{B}_{ij} = \begin{cases} \boldsymbol{I} - \boldsymbol{J}_i \boldsymbol{N}_{1,\dots,i-1} \boldsymbol{J}_i^\dagger, \ i = j \\ -\boldsymbol{J}_i \boldsymbol{N}_{1,\dots,j-1} \boldsymbol{J}_j^\dagger, \ \ i \neq j \end{cases} \quad (27)$$

Note: the upper-trianguluar block terms are zero as a result of (15).

Following the proof in Antonelli (2009), we use the Lyapunov function

$$V = \frac{1}{2} \vec{\sigma}^T \vec{\sigma} \quad (28)$$

which is positive for all non-zero $\vec{\sigma}$. It has a time derivative given by

$$\dot{V} = \vec{\sigma}^T \dot{\vec{\sigma}} \quad (29)$$

$$\dot{V} = -\vec{\sigma}^T \boldsymbol{A} \vec{\sigma} + \vec{\sigma}^T \boldsymbol{B} \frac{\partial \vec{\sigma}}{\partial t} \quad (30)$$

Looking at (26) and (27) we notice that from requirement (C), the off-diagonal elements of $\boldsymbol{A}$ and $\boldsymbol{B}$ are zero matrices of appropriate dimension. From requirement (A) and (D) $\boldsymbol{B}$ disappears entirely, giving

$$\dot{V} = -\vec{\sigma}^T \boldsymbol{A} \vec{\sigma}. \quad (31)$$

This means that the tracking problem has been reduced to the regulation problem. As described in Antonelli (2009) the block diagonal elements of $\boldsymbol{A}$ are positive definite from the requirement (B) and (C), and the task errors are asymptotically stable for any $\Lambda_i > 0$ with $i \in 1,\dots,k$. If requirement (D) holds for all tasks, not just the ones that have a time-varying aspect, then we also have $\boldsymbol{A} = \mathrm{diag}(\Lambda_1 \boldsymbol{I}_{m_1}, \Lambda_2 \boldsymbol{I}_{m_2}, \dots, \Lambda_k \boldsymbol{I}_{m_k})$.

## 3. EXAMPLES

To simplify Jacobian and pseudo-inverse calculation, the following examples were implemented in Python with CasADi (Andersson et al., 2018). CasADi is a symbolic and algorithmic differentiation framework for numeric optimization. The system is simulated with Euler's method and a timestep of 0.01 $s$.

We consider a highly redundant snake-like manipulator with 30 links of unit length with 30 revolute joints. The snake is rigidly attached at the base and moves in the plane. We have $\boldsymbol{q} = [q_1, q_2, \dots, q_{30}]^T \in \mathbb{R}^{30}$ with each subsequent joint angle defined relative the preceding joint. The forward kinematics to link $i$ is then given by

$$\boldsymbol{f}_{\boldsymbol{p}_i}(\boldsymbol{q}) = \begin{bmatrix} \sum_{j=1}^{i} \cos(\sum_{k=1}^{j} q_k) \\ \sum_{j=1}^{i} \sin(\sum_{k=1}^{j} q_k) \end{bmatrix} \quad (32)$$

and the partial derivatives of these are simply

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{p}_i}}{\partial q_l}(\boldsymbol{q}) = \begin{bmatrix} \sum_{j=l}^{i} -\sin(\sum_{k=1}^{j} q_k) \\ \sum_{j=l}^{i} \cos(\sum_{k=1}^{j} q_k) \end{bmatrix} \quad (33)$$

we denote the relative forward kinematics from link $i$ to link $j$ as $\boldsymbol{f}_{p_{i,j}}(\boldsymbol{q})$.

The first example emphasizes how feedforward of the task error derivative reduces the tracking error for tasks that are compatible. The second example shows two tasks that can be combined in the case of a regulation problem but will cause problems as a tracking problem. The third example shows a problem with discretization that is more evident in tracking tasks than in regulation tasks.

### *3.1 Example 1 - Effect of Feedforward*

We have three tasks forming three task errors:

$$\tilde{\boldsymbol{\sigma}}_1(t, \boldsymbol{q}) = \begin{bmatrix} 2\cos(0.1t) + 10 \\ 2\sin(0.1t) + 10 \end{bmatrix} - \boldsymbol{f}_{\boldsymbol{p}_{20}}(\boldsymbol{q}) \quad (34)$$

$$\tilde{\boldsymbol{\sigma}}_2(t, \boldsymbol{q}) = \begin{bmatrix} \cos(0.2t) + 2 \\ \sin(0.2t) + 2 \end{bmatrix} - \boldsymbol{f}_{\boldsymbol{p}_{25,30}}(\boldsymbol{q}) \quad (35)$$

$$\tilde{\boldsymbol{\sigma}}_3(t, \boldsymbol{q}) = q_{21} + q_{22} + q_{23} - \sin(t) \quad (36)$$

the first task is for the 20th link to follow a circle of radius 2 centered around $(10, 10)$, and the second is for the 30th link to follow a circle of radius 1 centered around $(2, 2)$ relative to the frame on the 25th link. The third tasks is for the sum of joints $q_{21}$, $q_{22}$ and $q_{23}$ to follow a sinusoidal signal. The tasks are annihilating, and fully represented in the necessary null-spaces, and we use gains $\Lambda_1 = \Lambda_2 = \Lambda_3 = 1$.

In Fig.1 we see the norms of the tasks w.r.t time both with and without feedback. Without feedforward, there is a persistent tracking error in all the tasks. In Fig.2 we see the smallest eigenvalue of $\boldsymbol{A}$ and the Frobenius norm of $\boldsymbol{B}$ w.r.t. time, note that $\boldsymbol{A}$ remains positive definite, and $\boldsymbol{B}$ is zero. As $\boldsymbol{B}$ is zero, the tracking tasks are fully represented in the null-space of the higher-priority tasks, and the task errors are asymptotically stable.

### *3.2 Example 2 - Incompatible Tasks*

A simple mistake during task design is to take what should be one task, and splitting it into two. Here we consider the case where we control the $x$ and $y$ position of the end-effector as two different tasks. This gives the task errors

$$\tilde{\sigma}_x(t, \boldsymbol{q}) = 2\cos(0.1t) + 10 - \sum_{j=1}^{30} \cos(\sum_{k=1}^{j} q_k) \quad (37)$$

$$\tilde{\sigma}_y(t, \boldsymbol{q}) = 2\cos(0.1t) + 10 - \sum_{j=1}^{30} \sin(\sum_{k=1}^{j} q_k) \quad (38)$$

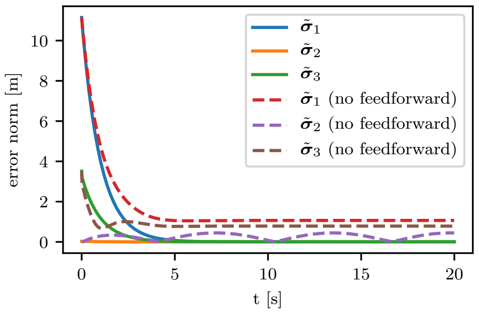


Fig. 1. Norm of task errors for example 1. With compatible tasks, feedforward removes tracking error.

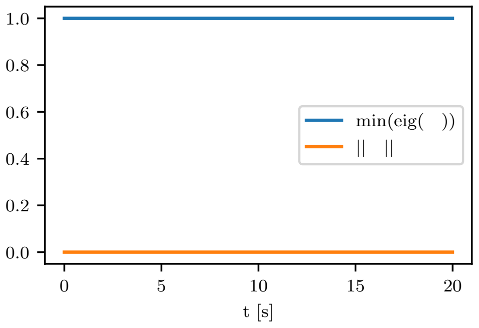


Fig. 2. Minimum eigenvalue of $\boldsymbol{A}$ and Frobenius norm of $\boldsymbol{B}$ for example 1. Only the case with feedforward is given as the no feedforward case is similar.

for which the partial derivatives are

$$\frac{\partial \tilde{\sigma}_x}{\partial q_i} = \sum_{j=i}^{30} \sin(\sum_{k=1}^{j} q_k) \tag{39}$$

$$\frac{\partial \tilde{\sigma}_y}{\partial q_i} = -\sum_{j=i}^{30} \cos(\sum_{k=1}^{j} q_k) \tag{40}$$

In Appendix A we show that as a regulation problem this is asymptotically stable, but not as a tracking problem. In Fig.3 we see the norm of the task errors. The high-priority $\tilde{\sigma}_x$ converges, but $\tilde{\sigma}_y$ does not as it is affected by the time-varying reference signal. In Fig.4 we see that the smallest eigenvalue of $\boldsymbol{A}$ is positive, and that $\boldsymbol{B}$ is non-zero.

### *3.3 Example 3 - Linearization*

CLIK approaches generally assume that $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{desired}}$, calculates $\dot{\boldsymbol{q}}_{\text{desired}}$, and then applies $\dot{\boldsymbol{q}}_{\text{desired}}$ to the robot for a time interval. This means that the time interval we apply $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{desired}}$ to the robot should be sufficiently small for the linearization assumption to hold. In Fig.5 we show the norm of the full task error vector in Example 1 for varying timestep sizes. In Fig.6 we show the same tasks but with a constant reference point instead of a reference trajectory.

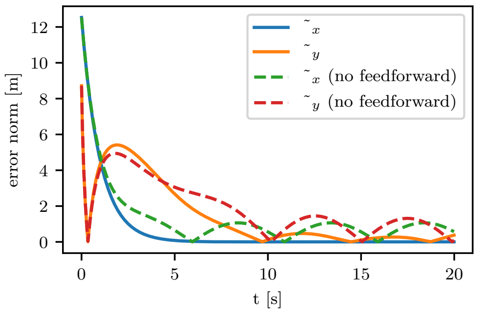


Fig. 3. Norms of the task errors for Example 2. Note that $\tilde{\sigma}_x$ converges but $\tilde{\sigma}_y$ does not.

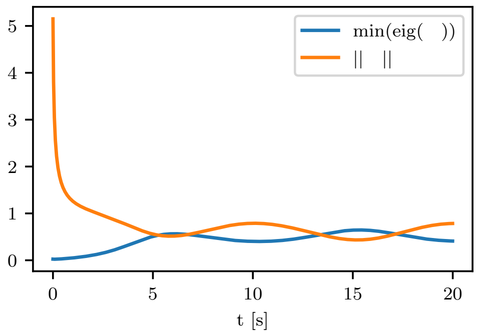


Fig. 4. Minimum eigenvalue of $\boldsymbol{A}$ and Frobenius norm of $\boldsymbol{B}$ for example 2. Only the case with feedforward is given as the no feedforward case is similar. Note that the minimum eigenvalue remains positive, but $\|\boldsymbol{B}\|$ is non-zero.

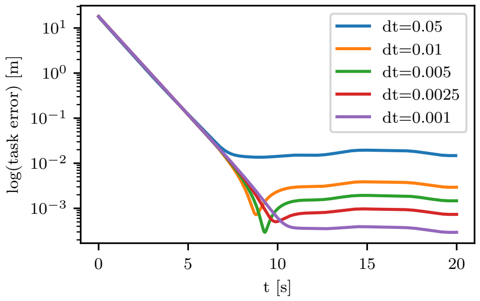


Fig. 5. Norms of the full task error vector from (34)-(36) for varying timestep lengths.

## 4. CONCLUSION

When performing tracking tasks, it is not as easy to combine tasks as when handling a regulation problem. In this article we presented a theorem showing sufficient conditions on the tasks to ensure asymptotic stability of the task errors for task-priority inverse kinematics for tracking problems. The new requirement when compared

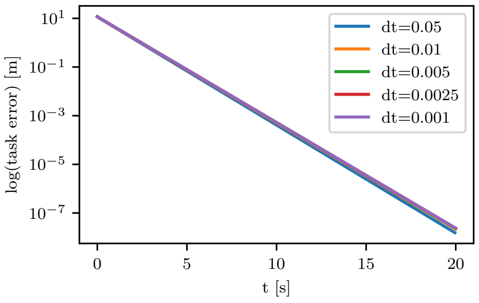


Fig. 6. Norms of the full task error vector from (34)-(36) for varying timestep lengths and a constant $t = 0$.

to the regulation problems is that tracking tasks are fully represented in the null-space of the previous tasks. This was shown with two examples, one showing compatible tasks where the tracking error was removed by the feedforward signal, and one showing a minimal example of incompatible tasks that could arise by simple user mistakes. The third example emphasized the dangers of linearization and discretization as a reminder that this becomes a larger issue when handling tracking problems. The errors observed in the third example are to be expected in other CLIK approaches where we assume the same linearity in the task error Jacobian. As the error observed is in the order of millimeters or even centimeters, this could negatively affect industrial use-cases such as seam-following as the error may exceed the desired tolerances if not accounted for.

The multiple task-priority inverse kinematics framework has been extended to set-based tasks in Moe et al. (2016). Extending the results of this article to set-based tasks will allow us to design control systems where we can ensure convergence while avoiding obstacles moving in known paths, or when the robot must remain inside a time-varying workspace.

The linearization issues may be addressed by extending the work of Falco and Natale (2011) to consider the tracking problem in the discretized version of the approach.

## REFERENCES

Aertbeliën, E. and Schutter, J.D. (2014). etasl/etc: A constraint-based task specification language and robot controller using expression graphs. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1540–1546.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2018). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*.

Antonelli, G. (2009). Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5), 985–994.

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3), 398–410.

Das, H., Slotine, J.E., and Sheridan, T.B. (1988). Inverse kinematic algorithms for redundant systems. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, 43–48 vol.1.

Falco, P. and Natale, C. (2011). On the stability of closed-loop inverse kinematics algorithms for redundant robots. *IEEE Transactions on Robotics*, 27(4), 780–784.

Moe, S., Antonelli, G., Teel, A.R., Pettersen, K.Y., and Schrimpf, J. (2016). Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results. *Frontiers in Robotics and AI*, 3, 16.

Sciavicco, L. and Siciliano, B. (1986). Coordinate transformation: A solution algorithm for one class of robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(4), 550–559.

## Appendix A. STABILITY OF EXAMPLE 2 AS A REGULATION PROBLEM

We have the partial derivatives (39) and (40) that form the Jacobians $\boldsymbol{J}_x \in \mathbb{R}^{1\times n}$ and $\boldsymbol{J}_y \in \mathbb{R}^{1\times n}$. We investigate the stability of the regulation problem by looking at $\boldsymbol{A}$. As $\boldsymbol{A}$ is lower-triangular, we are only interested in $A_{11}$ and $A_{22}$:

$$A_{11} = \Lambda_1 \tag{A.1}$$

and

$$A_{22} = \boldsymbol{J}_y(I - \boldsymbol{J}_x{}^{\dagger}\boldsymbol{J}_x)\boldsymbol{J}_y{}^{\dagger}\Lambda_2 \tag{A.2}$$

$$A_{22} = (1 - \boldsymbol{J}_y\boldsymbol{J}_x{}^{\dagger}\boldsymbol{J}_x\boldsymbol{J}_y{}^{\dagger})\Lambda_2 \tag{A.3}$$

We observe that

$$\boldsymbol{J}_y\boldsymbol{J}_x{}^{\dagger} = \frac{1}{\|\boldsymbol{J}_x\|^2}\boldsymbol{J}_y\boldsymbol{J}_x^T \tag{A.4}$$

$$\boldsymbol{J}_y\boldsymbol{J}_x{}^{\dagger}\boldsymbol{J}_x\boldsymbol{J}_y{}^{\dagger} = \frac{1}{\|\boldsymbol{J}_x\|^2\|\boldsymbol{J}_y\|^2}\left(\boldsymbol{J}_y\boldsymbol{J}_x^T\right)^2 \tag{A.5}$$

and that the tasks are not annihilating by design. By Cauchy-Schwarz, assuming $\boldsymbol{J}_y$ and $\boldsymbol{J}_x$ are non-zero (we avoid singularities), we have that

$$1 \geq \frac{\left(\boldsymbol{J}_x\boldsymbol{J}_y^T\right)^2}{\|\boldsymbol{J}_x\|^2\|\boldsymbol{J}_y\|^2} \tag{A.6}$$

which gives:

$$\mathrm{eig}(\boldsymbol{A}) = (\Lambda_1, \alpha\Lambda_2) \tag{A.7}$$

where $\alpha \geq 0$ as long as $\boldsymbol{J}_y \neq 0$, $\boldsymbol{J}_x \neq 0$ and the two are linearly independent. This means that given any arbitrary $\Lambda_1 > 0$ and $\Lambda_2 > 0$ the regulation problem is asymptotically stable. The tracking problem on the other hand is not. We have

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ -\beta & 1-\alpha \end{bmatrix} \tag{A.8}$$

where

$$\beta = \frac{\boldsymbol{J}_y\boldsymbol{J}_x^T}{\|\boldsymbol{J}_x\|^2} \tag{A.9}$$

If $\boldsymbol{J}_x$ and $\boldsymbol{J}_y$ are linearly independent, then $\beta = 0$ but $1 - \alpha \neq 0$. If $\boldsymbol{J}_x$ and $\boldsymbol{J}_y$ are dependent, then $1 - \alpha = 0$, but $\beta \neq 0$. The example shows how we can have stability of the regulation problem while not guaranteeing stability of the tracking problem with pseudo-inverse based closed-loop inverse kinematics.

# Paper 6 Interfacing KUKA Industrial Robots with ROS for Research and Education

M. H. Arbo, I. Eriksen, F. Sanfilippo, and J. T. Gravdahl, “Interfacing KUKA Industrial Robots with ROS for Research and Education,” submitted to Elsevier journal of *Mechatronics* January 2019.

# Interfacing KUKA Industrial Robots with ROS for Research and Education[☆]

M. H. Arbo[a,*], I. Eriksen[a], F. Sanfilippo[b], J. T. Gravdahl[a]

[a]*Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway*
[b]*Department of Science and Industry systems, University of Southeast Norway (USN), Post box 235, 3603 Kongsberg, Norway*

**Abstract**

In this work, an open-source ROS interface based on *KUKAVARPROXY* is proposed for control of KUKA robots. The interface allows for reading/writing variables of the controlled manipulators and is based on work previously introduced by our research group. This work also presents the process of investigating two available libraries for the interface, *BoostCrossCom* and *JOpenShowVar*, from the perspective of the end-user being in research and education. The interface is investigated in a use-case scenario often encountered in the industry, with sensors, tooling, and aging hardware.

To demonstrate the potential of the proposed interface, a systematic comparison with the commercial closed-source *Robot Sensor Interface* (RSI), a low-level control interface provided by KUKA for sensor feedback control of the robots using an external computer, is considered. This comparison provides an in-depth analysis of the communication latency expected in the system and an insight into the level of control that can be achieved with the KUKA robot system. The results highlight that even though the commercial interface is more reliable for feedback control tasks, the proposed open-source interface can approach similar behavior with a proper tuning of the override speed.

*Keywords:* Robot interface, Industrial Robots, Feedback Control, ROS

[☆]The work reported in this paper was based on activities within centre for research based innovation SFI Manufacturing in Norway, and is partially funded by the Research Council of Norway under contract number 237900.

[*]Corresponding author

*Email addresses:* `mathias.arbo@ntnu.no` (M. H. Arbo), `ivareri@stud.ntnu.no` (I. Eriksen), `filippo.sanfilippo@usn.no` (F. Sanfilippo), `jan.tommy.gravdahl@ntnu.no` (J. T. Gravdahl)

## 1. Introduction

### *1.1. Motivation and Outline*

Robotic manipulators perform an unimaginable variety of tasks. They are ubiquitous in modern industry and essential to meeting the future production demands of the world [1]. By using the same robot systems in research and education as in industry, the step from university to factory can be made surmountable. But as we do this, we must acknowledge the different requirements the different fields have. In industrial use cases there are strict requirements for uptime, safety, and repeatability, with few changes after initial installation and programming [2]. Each robot manufacturer usually provides a programming paradigm, with specialized hardware and software packages. In research use cases the software requirements are often bleeding edge, with unique hardware requirements. In education use cases the goal is a minimal representation of the robot showcasing the core principles of its use, with easy access for educators to control and survey the work of the students. These differences are often reflected in both the level of control the users have on the robot system, and the interfaces required. With the advent of Industry 4.0, the border between industry and research is shifting. New standards allow communications among different entities of an interconnected production system or system of systems, the requirement for industrial applications are becoming more demanding and similar to the research use cases with an increasing need for open software [3].

Most robot manufacturers provide two main methods of controlling the robot system: using a proprietary programming paradigm, or using an external computer. The proprietary programming paradigm is usually a part of the control computer and can be extended with commercial software and hardware. By using an external computer for control, the user can use custom software and hardware, at the cost of communication delay and requiring an interface between their software and the robot system. Access to the underlying control architecture in the robot system is often limited in both the proprietary programming paradigm and when using an external computer for control.

In this article we focus on the KUKA robot systems controlled by an external computer. KUKA produces robot manipulators ranging from collaborative lightweight robots such as the LWR IIWA 7 to the massive car-carrying KR 1000 Titan. A complete robot system by KUKA is comprised of the robot arm itself, a control cabinet e.g. KUKA Robot Controller 4 (KRC4), and a teach pendant referred to as the KUKA Control Panel (KCP). This article focuses on the KUKA KR 16 robot, and the KRC2 control cabinet. The robot is controlled from the control cabinet either manually using the teach pendant, or by running a KUKA Robotics Language (KRL) program. External axes attached to the robot can be controlled synchronously, or asynchronously, and the control cabinet can communicate with attached devices and external computers over Profibus or Ethernet.

While the KRL provides an interface that is easy to use in industrial applications, it is quite limited for research purposes. In particular, the KRL is tailored to the underlying controller and consequently, only a fixed, controller-specific set of instructions is offered [4, 5]. Extending the KRL to include new instructions and functionalities is an arduous task. Without a native way of including third party libraries in KRL, it is difficult to implement novel controllers and quickly connect to external input devices. To overcome such challenges, one of the most recognised and effective efforts that has come from the research community concerns the development of the Robot Operating System (ROS) [6].

ROS is an open-source middleware for writing robot software, it provides a message-passing structure for inter-process communication. Packages written in ROS are transferrable from one robot setup to another, and the large international userbase provides packages ranging from indoor navigation, to object identification, to reference frame calculation. The ROS community simplify software development through standards for data represtion, and both visualization and simulation tools. ROS-Industrial (ROS-I) is an open-source project aimed at extending ROS to new manufacturing application. ROS-I provides a standardisation of package structures for writing packages aimed at industrial robotics. In `ros_control`[7], ROS provides a framework for designing interfaces for controlling robot hardware from ROS. ROS provides support for different industrial robots including vendors like ABB, Adept, Fanuc, Motoman, and Universal Robots. Extensive research work has also gone into creating ROS drivers for KUKA robots using commercially available interfaces.

The main contribution of this work is an open ROS-based interface for KUKA industrial manipulators. Interfacing with industrial robots is not a trivial task, especially when the robots are to be used by everyone from first-year bachelor students to robotics researchers. This article discusses two different methods for controlling and communicating with a KUKA robot system from an external computer, an analysis of the communication latency expected in the system, and insight into the level of control that can be achieved with the KUKA robot system. The first of the two communication methods investigated uses the closed-source *Robot Sensor Interface* (RSI), a low-level real-time control interface provided by KUKA for sensor feedback control of the robots. The other method is communicating directly to the variables in a KRL program running on the KRC using the open-source *KUKAVARPROXY* (KVP) based *jOpenShowVar* [8] and *BoostCrossCom*[9]. This work is a continuation of [10] and [8], and investigates the KVP based control method for use in research and education.

The article is split into four sections. The first section gives the motivation and related research. Section 2 describes the robotics lab as well as the KRC2 and software used. Section 3 describes the process of creating the ROS interface, from evaluating the library to use, to designing the KRL program to run on the KRC2. Section 3 ends with a simple example of a grasp and place task executed with both the commercial and open-source interface. Section 4 gives a qualitative comparison of the interfaces and discusses the results for research and education.

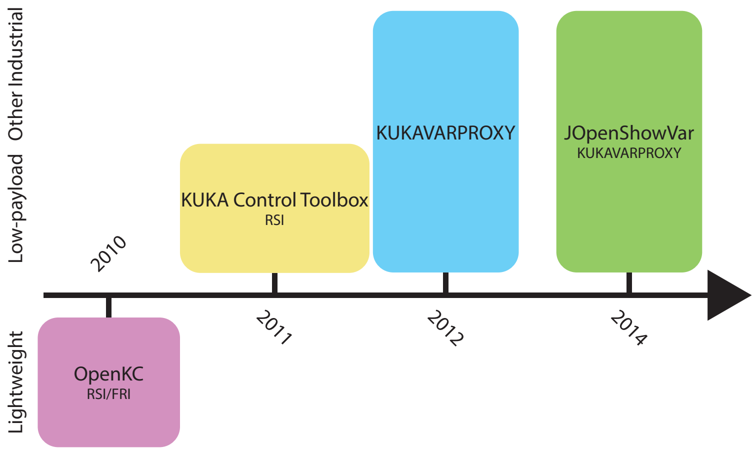


Figure 1: A time-line with recent related works. The works are categorized based on which class of KUKA manipulators they are intended to be used with. Where applicable, the underlying interface used is specified in small text under the name.

### *1.2. Related Research*

KUKA offers three interfaces for control and communication with a robot using an external computer: *Robot Sensor Interface* (RSI), *Ethernet KRL Interface* (EKI), and *Fast Research Interface* (FRI). RSI was used in this article and will be further discussed in the following sections. EKI is an interface intended for TCP/IP data communication between the computer and an external computer. As this data may include motion commands, it allows for motion control. EKI is generally less expensive than RSI, but not as viable for feedback control. FRI is a real-time interface supporting control modes from joint impedance control to joint position control, but it is only available for the KUKA lightweight manipulator series.

As shown in Fig.1, several research groups have explored the possibility of creating alternative software interfaces to KUKA robots. An open-source real-time control software for the KUKA lightweight robot, *OpenKC*, was presented in [11]. This software relies on either RSI or FRI, and is intended for use with POSIX compliant operating systems. *OpenKC* makes it possible to control the robot by using a simple set of routines that can easily be integrated into existing software. As a result, developers of robot applications can explore different software scenarios. However, the software is limited to the KUKA lightweight manipulator series, and use of RSI or FRI is required.

The *KUKA Control Toolbox*, introduced in [12], provides a collection of MATLAB functions for control of KUKA industrial robots. The underlying idea of the KCT is to offer an intuitive and high-level programming interface for the user. This toolbox is compatible with all small and low-payload KUKA robots that have six degrees of freedom (DOFs). The KCT runs on a remote computer connected to the KRC via TCP/IP. A multi-thread server runs on the KRC and communicates via RSI with a client whose job is to manage the information exchange with the manipulator. High transmission rates are guaranteed by this communication set-up, thus enabling real-time control applications. Nonetheless, as in the previous work, this approach requires the use of RSI.

KVP was developed by IMTS s.r.l. as a freely distributed server that runs on the Windows portion of the KUKA Robot Controller. The open-source communication library *JOpenShowVar*, was created by Filippo Sanfilippo et al. [8] as a Java based middleware for communication and control of the robot. It was created as an open-source alternative to current KUKA control packages. *JOpenShowVar* has been used in research on active heave compensation for offshore crane operations [13], robotic welding of tubes [14], sensorless admittance control in human-robot interaction [15], and in robot assisted 3D vibrometer measurements [16]. The open-source nature of *JOpenShowVar* allowed for the development of *BoostCrossCom*, a minimal C++ library for communicating with the KVP server. *BoostCrossCom* was developed by Eirik Njåstad for his master thesis [9] (results presented in [17]).

Even though *JOpenShowVar* and KVP have shown great potential, to the best of our knowledge, a ROS-based interface that works with all KUKA industrial robots without requiring FRI, EKI, or RSI has not been released yet.

## 2. Robot System and Setup

This section presents the robot system used in the experiments, and gives an overview of the software and hardware involved. This is to give insight necessary for reflecting on the behavior and applicability of the interface.

### *2.1. Thrivaldi*

The typical application categories for industrial robots include: handling, welding, assembly, painting, and processing. In all of these applications one encounters tooling and sensors. Similarly it is common to find external axes in the form of the manipulator being situated on a moving platform, or linear axes attached to the end-effector to increase the workspace of the robot. The robot system in this article features aspects that are common in such industrial use cases: aging hardware, robots with external axes, and tool and sensor communication. This prepares students for robot systems they may encounter in industry, and means that the interface investigations are applicable to a wide range of robot setups.

The lab, see Fig.2, is situated at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) and consists of two 6 degrees-of-freedom KUKA KR 16 robots, where one is attached to a GÜDEL gantry crane giving it 9 degrees-of-freedom. The lab was named after Thrivaldi, a 9 headed giant from Norse mythology. With its impressive size, and

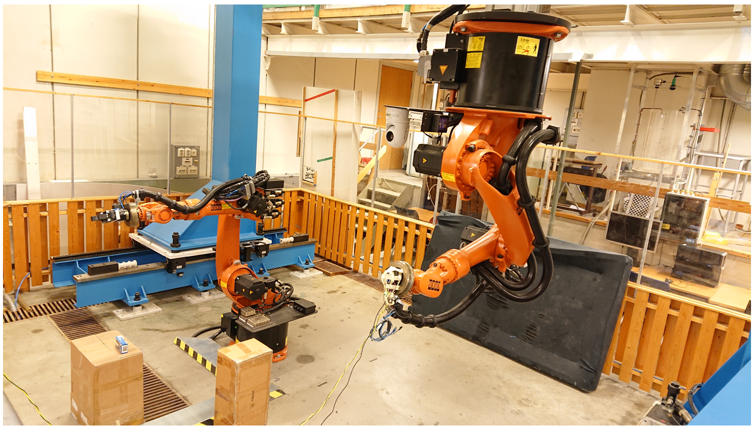


Figure 2: Thrivaldi with a floor mounted KR 16 with a pneumatic SCHUNK gripper attached (RH9010), and a gantry mounted KR 16 with an IMU attached.

large workspace, the lab is a good motivator for young students and a capable tool for researchers.

Each robot is controlled with a KRC2 cabinet. The gantry crane is connected to the KRC2 and set up as synchronous external axes controlled directly by the cabinet. Both robots have a SCHUNK FTC-50-80 force/torque sensor attached, and a pneumatic toolchanger. The floor mounted KR 16 has a SCHUNK RH9010 pneumatic parallel gripper attached, controlled with an SMC EX250 pneumatic controller connected to the KRC2's over Profibus. The github project describing the lab setup is available [18] and includes ROS drivers for controlling synchronous external axes with an RSI interface.

### *2.2. Software and Hardware*

To understand how the KVP and RSI-based control interfaces differ in how they interact with the controller cabinet, we must give an overview of the software and hardware commonly found on KUKA robot systems. Specifically we introduce: the KUKA Robot Controller (KRC), the KUKA Robot Language (KRL) interpreter, the KUKA Simple Programmable Logic (SPS) interpreter, the KUKA Robot Sensor Interface (RSI), and *KUKAVARPROXY* (KVP). This section focuses on the hardware in Thrivaldi but is generalizable for other industrial KUKA robot systems requiring control from an external computer. For a more in-depth description of the KUKA software and hardware ecosystem, and programming within the propriatery KRL paradigm, we refer the reader to the documentation from KUKA.

### *2.2.1. KUKA Robot Controller*

The KRC2 encompasses the power supply, servo controllers, control computer, I/O, etc. It is fully capable of controlling complex robot systems on its own with KRL programs and software programmable logic controllers. The control computer in the KRC2 is a standard x86 computer with a single core Intel Celeron CPU. The KUKA System Software (KSS) is the software collection running on the control computer responsible for all robot operations. KSS uses VxWin, a KUKA specific real-time operating system based on VxWorks that runs both VxWorks and Windows XP. More modern versions of the KRC such as the KRC4 has multi-core CPUs and more recent versions of Windows.

The KRC2 has an interpolation cycle (IPOC) of 12 ms. Within each IPOC, the KRL interpreter runs, I/O devices are updated, the software programmable logic controller (*SPS*, from *speicherprogrammierbare steuerung* in German) runs, and Windows tasks are executed. It is important to note that KRL programs and I/O are of higher priority than SPS and Windows tasks. As robot control is the goal the control computer prioritizes execution of any running KRL program over handling Windows requests.

### *2.2.2. KRL Interpreter*

KRL is KUKA's Pascal-based proprietary programming language. It has robot-oriented commands and data types, specified both for motion in Cartesian and joint space. The KRL interpreter executes the sequential commands defined in a program, performing inverse kinematics and motion planning where necessary, writing and reading variables when needed, and communicating with analog and digital I/O. To understand the KVP based interfaces, we only require the system variables `AXIS_ACT`, `ADVANCE`, `OV_PRO`, and the motion command `PTP`. `AXIS_ACT` is the current joint coordinates of the robot. `ADVANCE` defines how many commands ahead the motion planner should look when performing path smoothing. `OV_PRO` is the override speed percentage, the percentage of maximum speed permissible when executing a motion command. Note that the override speed is not necessarily the speed that will be used, but the speed limit in the motion planner. `PTP` is a point-to-point motion command to either a pose defined in Cartesian space or joint space. The `PTP` command executes a trapezoidal motion between the current pose to the desired pose. If the option `C_PTP` is supplied to the `PTP` command, and `ADVANCE>0`, the motion planner will approximate the motion to the pose, and start to move towards the next desired pose as soon as the robot is sufficiently close to the current desired pose. This smooths the motion, making it more efficient and faster, at the cost of positional accuracy. KRL supports defining workspaces that can constrain motion commands and be used to ensure that certain functions are only executed when the end-effector is in a particular workspace.

### *2.2.3. SPS Interpreter*

Simple programmable logic is intended to be run in the SPS interpreter. A single KRL based SPS program is set up as the main program that starts at controller boot, and runs as long as the KRC2 is powered. SPS cannot move synchronous axes such as the gantry crane in Thrivaldi, but I/O such as the valve controlling the gripper, and asynchronous external axes commonly attached to the end-effector of the robot. In Thrivaldi SPS is used for controlling the toolchanger, pneumatic gripper, and force-torque sensor as these are attached to the control cabinet via Profibus.

### *2.2.4. Robot Sensor Interface*

The Robot Sensor Interface (RSI) is a commercially available software package for KSS intended for sensor assisted motion and data exchange. The idea is to use external sensors to correct the position of the robot while it is following a trajectory. RSI lets us directly communicate with the VxWorks portion of the KRC2 using TCP or UDP and the connection is dropped if too many responses are late or absent. RSI will try to apply any position corrections within the IPOC it receives them. Communication with RSI from an external computer is done using KUKA.Ethernet RSI XML. The messages are transmitted as XML strings, and the data objects to transmit can be configured from KRL.

In the `kuka_experimental` package, RSI has been used to create an interface to control KUKA robots via ROS. The ROS interface uses joint position corrections to control the robot. The RSI position corrections are intended for minor joint or Cartesian position corrections and work at a lower level than the KRL interpreter. This means that it has a separate configuration for the maximum position corrections (effectively the override speed), and neither adheres to workspace limitations, nor perform trapezoidal motion between current and desired position. For ROS independent movement latency testing in Sec.3.1.2, a barebones C++ interface was created from the XML header files in `kuka_experimental`. A modified version of the package that supports external axes can be found on the project repository[18].

### *2.2.5. KUKAVARPROXY*

KUKAVARPROXY (KVP) is a multi-client server that runs in Windows on the KRC2 and gives TCP/IP access to external computers. KVP communicates with the KRL interpreter using the *CrossCom* library, and can read and write to global variables in the interpreter. To move the robot we must have a KRL program running on the KRC2 with a loop that executes a motion command with a KVP writable global variable. This is discussed in Sec.3.2.2. Although *CrossCom* allows for interaction with the real-time control processes, the KVP server runs in the lower-priority Windows OS, which introduces communication latency. This latency is investigated in Sec.3.1.1. There are two main libraries available for communicating with KVP. *JOpenShowVar*, presented in [8], is a Java library with classes for all KRL variable types. *BoostCrossCom* [9] is a minimal C++ library inspired by *JOpenShowVar*, but is not as feature rich as *JOpenShowVar* and does not provide classes for all KRL variable types.

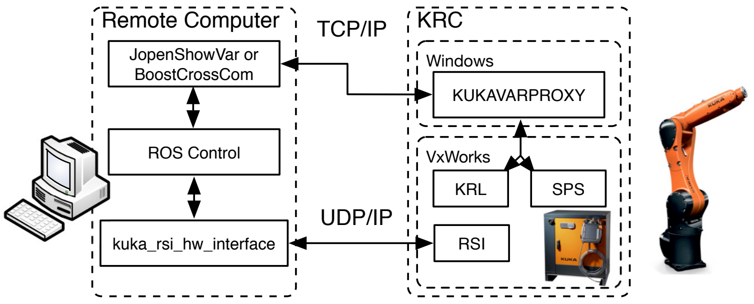


Figure 3: We have two paths of communication from an external computer running ROS to the robot hardware. One going via RSI to VxWorks, and one going via KVP through Windows to KRL and SPS. Both paths can be used simultaneously.

## 3. ROS Interface

In this section we describe the process of evaluating which of the *JOpenShowVar* and *BoostCrossCom* libraries is best suited as the basis for a KVP based ROS interface.

`ros_control` hardware interfaces are used to communicate between ROS and the robots. `ros_control` provides a hardware abstraction layer and generic controller plugins that can be activated and deactivated during runtime. This allows for controlling the robot using a stream of messages in a feedback situation, then switch to serving it a trajectory using a cancellable ROS action. Examples of controllers include the `joint_state_controller` that publishes joint states, and the `joint_position_controller` for giving joint position commands to the robot. RSI is well established in the ROS-I community with the `kuka_experimental` package, and in this section it will serve as the baseline on which the *JOpenShowVar* and *BoostCrossCom* libraries will be evaluated. A visualization of how the two interfaces have different pathways to control the robot is shown in Fig. 3. The KVP based interface communicates with the KVP server running on the Windows portion of the KRC, it reads and writes to global variables available to the SPS and KRL interpreters. The RSI based interface communicates directly to the real-time operating system on the KRC, allowing for low-level control that ignores the motion planner and the override speed in the KRL interpreter. It has its own configuration file that is loaded in the KRL program one needs to run to establish a connection from the KRC to the external computer.

### *3.1. Evaluating the libraries*

*JOpenShowVar* is implemented in Java, with full support for all KRL variable types. *BoostCrossCom* is implemented in C++, and supports a small set of KRL variable types. Qualitatively, C++ support is more mature in ROS than Java support, but JOpenShowVar is a more mature library. For a more quantitative comparison, a test was performed for the access time, and for the movement latency when using these two libraries. Timing was performed using the Boost `cpu_timer` class in C++, and `System.nanoTime` method in Java. The tests were performed on a computer with an Intel Xeon CPU E5-1650 v3 running Ubuntu 16.04 with ROS Kinetic Kame.

### *3.1.1. Access time*

The access time is a communication latency from ROS to the hardware resulting from the KVP server running on the lower-priority Windows side of the KRC, see Fig.3. The access time refers to the time to access the global variables in the interpreters from an external computer. Filippo et al. reported in [8] an average access time of approximately 5 ms when writing to an `E6AXIS` variable using *JOpenShowVar* on a KRC4. We investigate the access time for both *JOpenShowVar* and *BoostCrossCom* on a KRC2 when reading a variable (READ), writing to a variable (WRITE), and for a write-copy-read strategy (SYNC).

We implement SYNC as writing to `VAR1` and then reading from `VAR2` until it equals `VAR1`. In the main SPS, the value of `VAR1` is written to `VAR2`. As the only thing running on the KRC is the SPS and Windows, and Windows is of lower priority than the SPS, the SYNC strategy becomes an estimate of the delay from a variable is written to until it becomes available for the KRL interpreter. The SYNC command is representative of using the SPS for real-time processing and access to KRC global variables, tooling, or state-machines.

The tests were performed for both a composite datatype (`E6AXIS`) and a single data type (`INT`) as these are our main use cases. To see if multiple connections affect the access time READ and WRITE were performed with 1 and 5 threads. SYNC was performed both with 1 thread, and with 1 thread and 4 threads running READ to simulate network traffic. Each test was performed 50000 times. For the setup available, 5 connections were the most that could be reliably used with the KVP server.

The access time statistics of the full set of tests are given in Appendix Appendix A. The libraries have, for the most part, comparable performance. `INT` is faster to both READ and WRITE than `E6AXIS`, and having 1 connection is faster than having 5 connections. READ and WRITE for an `E6AXIS` takes approximately 3 ms each for both libraries with 1 connection. For 5 connections READ takes approximately 11 ms and WRITE takes approximately 16 ms. For 1 connection SYNC takes approximately 24 ms for `E6AXIS` using *BoostCross-Com*, and approximately 30 ms with 5 connections. Using *JOpenShowVar* with 1 connection SYNC takes approximately 24 ms for `E6AXIS`, but approximately 35 ms with 5 connections.

To visually compare the results we use a cumulative percentage of the tests with respect to the access time. As seen in Fig.4, *BoostCrossCom* is more reliable than *JOpenShowVar*. The figure shows that one can assume 80% of the `INT` SYNC tests to have less than 30 ms access time with *BoostCrossCom*, but only 37% with *JOpenShowVar*. In general with 5 connections, one can expect

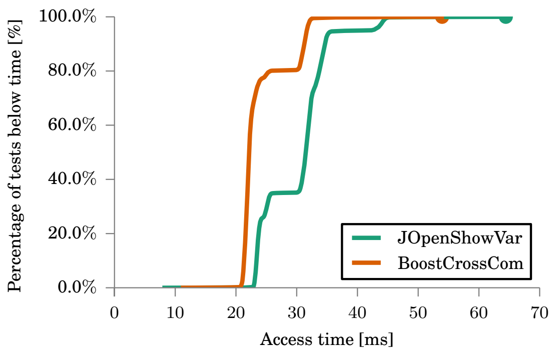


Figure 4: Comparison of the percentage of tests that had a SYNC latency under a certain time for *JOpenShowVar* and *BoostCrossCom*. The tests were performed with 5 threads transmitting the `INT` variable. The final dot denotes the maximum latency.

SYNC to take around 24-40 ms for `INT` datatypes, and 24-48 ms for `E6AXIS` datatypes.

*3.1.2. Movement Latency*

For feedback control, access time is secondary to the movement latency. The movement latency is the time from a write command is issued until the robot starts moving. It can adversely affect feedback control situations by introducing a delay that can lead to instability. This section investigates whether the movement latency differs from the access latency experienced with `SYNC`.

To get a measurement that is independent of the library used, we placed an Arduino Micro with an MPU-6050 IMU on the end-effector of the gantry-mounted robot. The time from the write command is issued until the Arduino responds that the IMU has sensed a change in acceleration is recorded as the measurement time. This measurement is independent of the internal behavior of the interface. To further separate the measurement from the interface, listening for the IMU was done in a separate thread.

Using a ROS independent barebones C++ RSI interface based on the RSI interface in ROS, we establish a baseline for the movement latency we can achieve with commercially available software packages.

For the KVP based libraries, the robot moved joint A5 30° up and down with an override speed of 100%. This uses the `PTP` command in KRL which exhibits trapezoidal motion on the joint angles (the KRL program running on the KRC is given in Listing 1). For RSI, a simple trapezoidal motion was implemented with similar acceleration. The robot was given 10 seconds to stabilize after each motion had executed to ensure that vibrations caused by the motions would not affect the subsequent measurements. This also means that for each execution

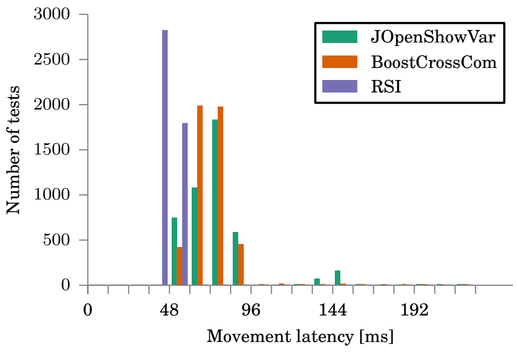


Figure 5: Movement latency for the three different interfaces. Each bin of the histogram is the length of one IPOC (12 ms). Note that RSI is only in the 4th or 5th IPOC whereas both JOpenShowVar and BoostCrossCom have cases with longer delay.

the robot controller must overcome the static friction in the joint. For each library the test was performed 5000 times.

Fig.5 shows the results as a histogram where each bin is the length of one IPOC (12 ms). RSI consistently uses 4-5 IPOCs to overcome the static friction in the joint, and the KVP libraries use longer. Statistics of the results are given in Tab.1. The average movement latency of *BoostCrossCom* and *JOpenShowVar* are comparable, but *BoostCrossCom* outperforms *JOpenShowVar* when it comes to variance. Both libraries had tests in the hundreds of milliseconds range, but *JOpenShowVar* had significantly more. As they did not differ as much in the SYNC experiments, this could be a result of differing behavior of *RxTx* (used in Java) and POSIX (used in C++) for communicating with the Arduino.

The KVP based interfaces have 2-3 IPOCs longer movement latency when compared to RSI. This is similar to what was observed in the SYNC experiments with one connection, and it is expected that the movement latency may increase when more connections are used.

Table 1: Movement Latency for 5000 Tests

| Library | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|
| *BoostCrossCom* | 49.98 | 268.72 | 76.15 | 77.62 | 18.59 |
| *JOpenShowVar* | 51.16 | 402.73 | 84.67 | 78.58 | 38.24 |
| *RSI* | 31.10 | 55.82 | 46.86 | 46.87 | 3.88 |

### *3.2. Implementation*

Three considerations from the previous sections are: the aging hardware in the lab (KRC2) allows at most 5 connections to KVP, *BoostCrossCom* has marginally lower variance in access time and movement latency, and ROS has a more mature support for C++ than Java. Because of this, the ROS interface was implemented in C++ using *BoostCrossCom*. In this section we describe the process of designing a KVP based ROS interface using *BoostCrossCom* that minimizes the effect of our lab limitations, and investigate design considerations such as `C_PTP` and `OV_PRO` for the KRL program running on the KRC.

### *3.2.1. ROS Hardware Interface*

The most important limiting factor for the implementation of the interfaces is the number of connections that can be established with the KVP server. To mitigate this we developed the interface with a set of nodes.

Following the ROS-I conventions, the hardware interface is available in the `kuka_kvp_hw_interface` package [19]. It implements a set of nodes that can be activated depending on what is needed. Motion control is exposed via the `kvp_joint_command_node` that contains generic `read` and `write` commands for the joint pose. This allows it to be used with the `joint_position_controller` plugin for streamed message control, and the `joint_trajectory_controller` plugin for trajectory actions. The node establishes two connections to the KVP server, one for reading and one for writing the joint commands.

If one only desires to read the joint states, the `kvp_joint_state_node` achieves this without interfering with any running KRL program on the robot. This node establishes a single connection to the KVP server.

`kvp_variable_interface_node` exposes ROS services for reading or writing to any global variables referenced by name. This gives access to control and monitoring of tooling and other devices attached to the KRC2. The connections to the KVP server are established in the services, and are therefore meant for intermittent transmission of signals.

By design, the `kvp_variable_interface_node` has been made to cause the least amount of network load by only establishing the connections when needed. This is an attempt at reducing the increase in access latency caused by multiple connections running simultaneously.

The code is publically available at [19].

### *3.2.2. KRL Program and Tracking Delay*

The KVP based ROS interface relies on a KRL program running on the KRC. As noted in [8], this program can be designed to suite the needs of the particular application. To generalize the interface for a varied use and to minimize how much students will have to interact with the KRC, we implement the KRL program as a simple loop that moves the robot to the desired pose in joint space (see Listing 1). There are three design parameters to the `PTP` command: `OV_PRO` that governs the override speed, `ADVANCE` that governs the look-ahead motion planner, and `C_PTP` that activates path smoothing. To investigate how they affect the motion profile, a simple ROS node publishing a sinusoidal signal on joint A3 was created in Python. The desired joint angle is

$$q_{A3,\text{des}}(t) = A\sin(\omega t) \tag{1}$$

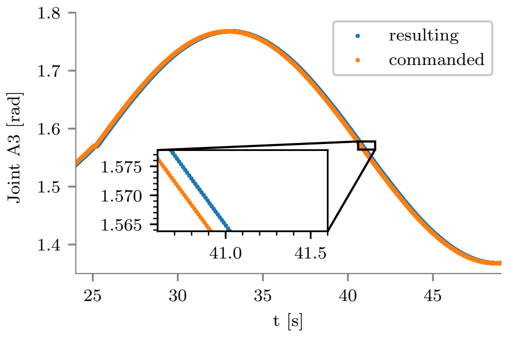


Figure 6: Joint A3 moving with a sinusoidal motion using RSI. Note that there is an approximately 120 ms delay from a command is published until it is reported that the robot achieves the desired joint position.

where $A = 0.2$ rad, and $\omega = 0.2$ rad/s.

The access time with KVP affects the current investigation as stochastic noise on the measured joint angles, but the major periodic behavior of the reported joint states are indicative of the effect of the design parameters.

In Fig.6 the RSI interface was used to establish a baseline performance. There is an approximately 120 ms tracking delay from a command is given until the robot achieves the same joint angle and it is reported in ROS. These results are similar to that of Lind et al. in [20] where tracking delay was tested with RSI without ROS. We therefore assume that any latency caused by the ROS communication stack and the reported timestamps of the rosbag to be negligible with respect to the tracking delay.

In Fig.7 the KVP based interface was used with 100% override speed, an `ADVANCE` of 1, and no `C_PTP`. The robot moves in a stop-and-go motion. As the override speed is very high, the robot performs a trapezoidal motion to the desired pose and then stops when the pose is reached. After this the robot requires some time to start a new motion, resulting in a stop-and-go motion.

With `C_PTP` active, the robot will start moving to the next pose as the new commanded pose is updated. However, as can be seen in Fig.8, with an override speed of 100%, the robot has moved to the next commanded pose by the time the new commanded pose is written. This causes an even more severe stop-and-go motion.

In Fig.9 `C_PTP` is activated and the override speed is set to 30%. This is approximately the speed at which the robot is required to move by the commanded signal, and we see that we can achieve behavior that is close to RSI, exhibiting approximately 150 ms delay.

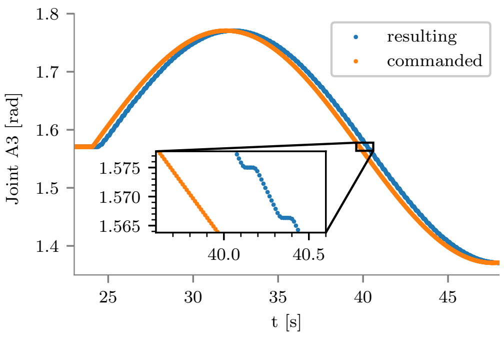


Figure 7: Joint A3 moving with a sinusoidal motion using KVP at 100% override speed without `C_PTP`.

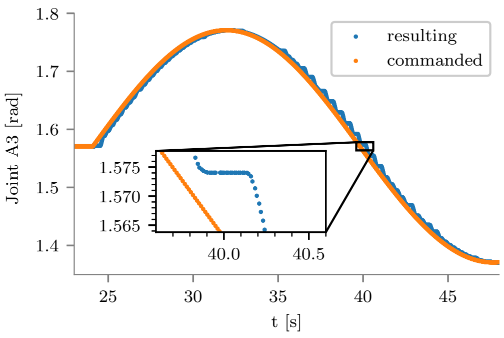


Figure 8: Joint A3 moving with a sinusoidal motion using KVP at 100% override speed with `C_PTP`.

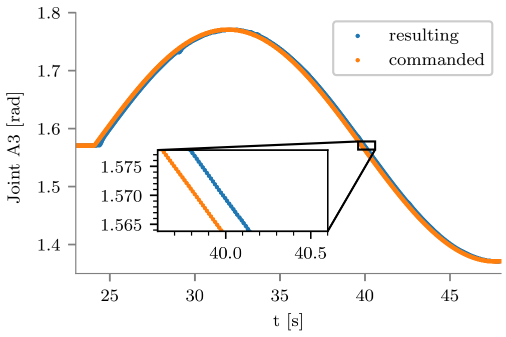


Figure 9: Joint A3 moving with a sinusoidal motion using KVP at 30% override speed with `C_PTP`.

Listing 1: KRL Program for KVP-Based Control

```
DEF kukavarproxy( )
INI
$ADVANCE = 1;
AXIS_SET = $AXIS_ACT;
LOOP
PTP AXIS_SET C_PTP
ENDLOOP
END
```

### *3.3. Access Time and Feedback Control*

For feedback control situations where we desire joint velocity control the stochastic delay caused by the access time may adversely affect our control. In this example we control joint A1 by following the procedure by Filippo et al. in [8] such that

$$q_{A1,des}(t_{k+1}) = \hat{q}_{A1}(t_k) + \dot{q}_{A1,des}\Delta_t \tag{2}$$

where $\hat{q}_{A1}(t_k)$ is the reported joint angle, $\dot{q}_{A1,des}$ is the desired joint velocity, and $\Delta_t = t_{k+1} - t_k$ is one IPOC (12 ms). If run in an open-loop manner, we can replace the reported joint angle with the previous $q_{A1}(t_k)$ that was sent to the robot.

To investigate the effect of the stochastic delay and design parameters in feedback control scenarios, we want to track

$$q_{A1,des}(t) = A\cos(\omega t) \tag{3}$$

where $\boldsymbol{A} = 0.349$ rad (corresponding to 20°) and $\omega = 0.314$ rad/s (corresponding to 2 Hz). With velocity control this means we send

$$q_{A1,des}(t_{k+1}) = \hat{q}_{A1}(t_k) - \omega A \sin(\omega t_k) \tag{4}$$

to the robot. This was implemented as a simple ROS node in Python.

In Fig.10 the RSI interface was used. Again there is a 120 ms tracking delay that is inherent to the system, but the desired joint angle is tracked quite well and the results do not drift.

In Fig.11 the KVP interface was used with 100% override speed. The stochastic delay affects the signal, and the error caused by this is integrated. Later on in the experiment, the external computer is synchronized with respect to the control cabinet and tracking is acceptable albeit shifted. The KVP interface exhibits a similar 120 ms tracking delay. As `C_PTP` is active, the robot does not span the whole of the desired curve and the resulting amplitude is slightly lower than the desired.

In Fig.12 the KVP interface was used with 100% override speed but the external computer is not synchronized with respect to the control cabinet. This causes the robot to not find the new desired joint angles ahead of time, and the motion becomes more stop-and-go, resulting in an integration of the error and worse tracking of the desired behavior.

In Fig.13 the KVP interface was used with 50% override speed. For the most part this works well, however, the lower override speed does not guarantee that synchronization will occur. This means that the motion becomes stop-and-go and we integrate the error. Lower override speeds were tested but exhibited similar results.

To mitigate the effect of the override speed, automatic tuning was implemented where the KVP variable interface was used to set the override speed automatically before each joint position command is sent. The override speed was chosen according to

$$\texttt{OV\_PRO} = \frac{\dot{q}_{A1,des}}{\dot{q}_{A1,max}} \tag{5}$$

where $\dot{q}_{A1,max} = 2.722\text{rad/s}$ according to the KR 16 manual. In Fig.14 we see the same experiment with automatic tuning of the override speed. The automatic tuning helps mitigate the severity of the stop-and-go motion, but there is a access latency in setting the override speed, and stop-and-go motion does occur.

### *3.4. Closed-Loop Inverse Kinematics Example*

In this example we want the end-effector to follow a 3D Lissajous trajectory defined by

$$\boldsymbol{p}_{des}(t) = 0.3 \begin{bmatrix} \sin(n_x \omega t) - 1.4 \\ \sin(n_y \omega t) + 0.7 \\ \sin(n_z \omega t) + 1.0 \end{bmatrix} \tag{6}$$

where $\omega = 0.02$, $n_x = 5$, $n_y = 2$, and $n_z = 3$. The trajectory w.r.t. the robot in its initial position is visualized in Fig.15. The error between the current position

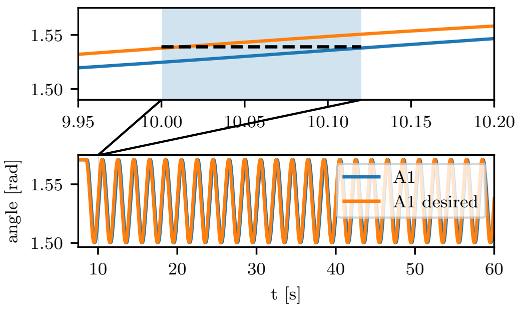


Figure 10: Velocity-resolved sinusoid with the RSI interface with 120 ms tracking delay.

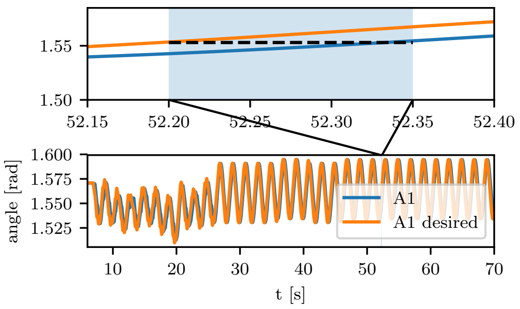


Figure 11: Velocity-resolved sinusoid with the KVP interface and an override speed of 100% with 150 ms tracking delay. From $t = 45$ we synchronize and follow the sinusoid well.

and the desired position is described by

$$\boldsymbol{e}(t, q) = \boldsymbol{p}(q) - \boldsymbol{p}_{des}(t) \tag{7}$$

which we desire to converge to zero.

To do this we apply CASCLIK[21], a CasADi-based[22] closed-loop inverse kinematics Python framework. The framework allows for formulating controllers for multiple constraint-based tasks and solves them either using the Moore-Penrose pseudoinverse, or as constraints to an optimization problem. In this example we apply the quadratic programming controller, and CASCLIK formulates the constraint

$$\boldsymbol{J}\dot{\boldsymbol{q}}_d = -K\boldsymbol{e} - \frac{\partial \boldsymbol{e}}{\partial t} \tag{8}$$

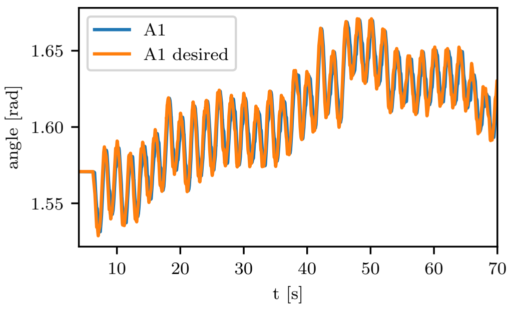


Figure 12: Velocity-resolved sinusoid with the KVP interface and an override speed of 100%, but synchronization does not occur and the error from the stop-and-go motion is integrated.

and the cost

$$c = \dot{\boldsymbol{q}}_d^T \dot{\boldsymbol{q}}_d \tag{9}$$

where $\boldsymbol{J}$ is the task Jacobian automatically generated from the KR 16 URDF, $\dot{\boldsymbol{q}}_d$ is the desired joint velocity that will be applied, $K = 50$, and $c$ ensures that the resulting joint velocity remains bounded. CASCLIK was chosen as it has a velocity-resolved ROS node that applies the same method as in the previous section for velocity-resolved control but is intended for closed-loop position control.

In Fig.16 we see the Euclidean norm of the error when for four different experiments: using the RSI interface, using KVP with 100% override speed and achieving synchronization, using KVP with 100% override speed but not achieving synchronization, and using KVP with automatic tuning. If synchronization is achieved the KVP with 100% override speed is close to the tracking error of the RSI interface, but access delay may cause stochastic errors that makes us lose this synchronization. With automatic tuning of the override speed we can reduce the occurrences of the stop-and-go motion.

As CASCLIK attempts to achieve exponential convergence, the errors caused by the stop-and-go motion have an exponential convergence to back to the minimum tracking error. The minimum tracking error is a result of not having an integrating effect in the controller, from the tracking delay in the interface, and from the linearization assumption inherent in (8).

## *3.5. Benefits of Abstraction: An Education Example*

In this example the robot is to move a stapler from one cardboard box to another using either RSI for motion, or KVP. The robot moves between four points defined in joint space, stopping momentarily above the cardboard boxes to release and grasp the stapler. With RSI, the trapezoidal motion profile in joint space between any two points is created by the external computer and

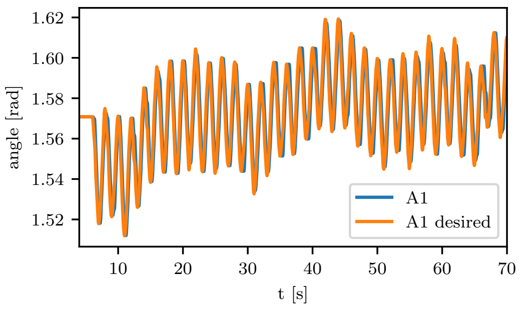


Figure 13: Velocity-resolved sinusoid with the KVP interface and an override speed of 50%. Synchronization is not achieved but the stop-and-go motion is reduced.

commanded as small angle corrections. With KVP, only the end position is commanded and system relies on the internal trapezoidal motion planner. The gripper and toolchanger are controlled using SPS and the `kvp_variable_interface`. To ensure a student cannot open the gripper while the robot is in motion, the gripper can only be activated when the end-effector is in designated workspaces above the boxes.

In Fig.18 the Cartesian motion of the end-effector with respect to time is given for when the task is performed with the KVP based interface. Both the robot and gripper is controlled using the KVP interface. In Fig.19 the Cartesian motion of the end-effector with respect to time is given for when the task is performed with the RSI based interface. In this case the motion is commanded using RSI and the gripper is controlled using ROS services from the `kvp_variable_interface_node`. Note that with only the KVP based interface, the motion is slightly smoother and slower as the KRL program is running with an override speed of 30% and uses the internal trapezoidal motion planner of KUKA. The RSI based interface uses a trapezoidal motion created in the external computer that does not exactly match the KVP based interface. The overall curvature of the two are similar, and the example demonstrates usage of both KVP and RSI at the same time for motion and tool control.

As the KVP based interface inherently uses the `PTP` command in KRL, the students will learn to program similar to what one might encounter in the industry. With the RSI based interface, `OV_PRO` does not affect the execution speed, but with the KVP based interface it does. This means that an educator can control the execution speed of anything the students may implement by simply adjusting the value from the pendant.

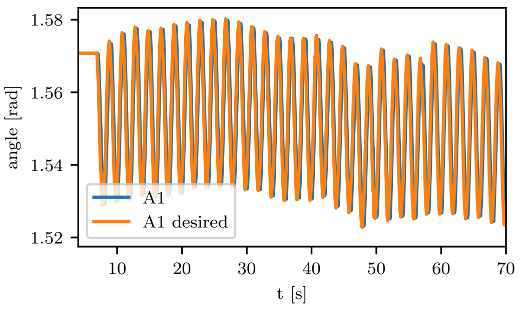


Figure 14: Velocity-resolved sinusoid with the KVP interface and automatic tuning of the override speed. Synchronization is not achieved but the stop-and-go motion is further reduced.

## 4. Discussion

The qualitative comparison between the two interfaces investigated is summarised in Tab.2. RSI is a commercial real-time interface which demands care and consideration of the programmer, both in terms of safety handling and keeping the real-time communication requirements. The KVP based interface is an open-source interface with natural limitations in how fast and accurate one can expect the control to be, both in terms of timing reliability and path accuracy when `C_PTP` is active.

*JOpenShowVar* and *BoostCrossCom* are comparable libraries for communicating with the KVP server running on the KRC, but *BoostCrossCom* is favorable in two regards. *BoostCrossCom* has lower variance in the access time and movement latency and as the rosjava client library is still maturing, the C++ based *BoostCrossCom* is easier to use with ROS.

Closed-loop inverse kinematics should be applied when using the KVP interface. The stochastic delay caused by the access time and the potential for stop-and-go motion in the motion planner of the KRC2 makes it difficult to rely on the convergence of any joint velocities applied to the robot. Lind et al.[20] attributes the 120 ms tracking delay of the RSI interface to motion buffers in the system. Although KVP uses point-to-point motion in a different manner than RSI, it also exhibits a minimum tracking delay of the same order of magnitude. To reduce the stop-and-go motion would require more accurate modeling of the internal motion planner, which can be easier to do now that the KVP allows inspection of the joint variables while a KRL program is being executed.

The different behavior of RSI and KVP based ROS interfaces suggests the possibility of different usage scenarios. The KVP based interface is closer to programming in KRL. It provides the same layer of abstraction from motion profile planning that may stop novice users of the robot system, and it does not require timely responses to the external computer. This means that one can

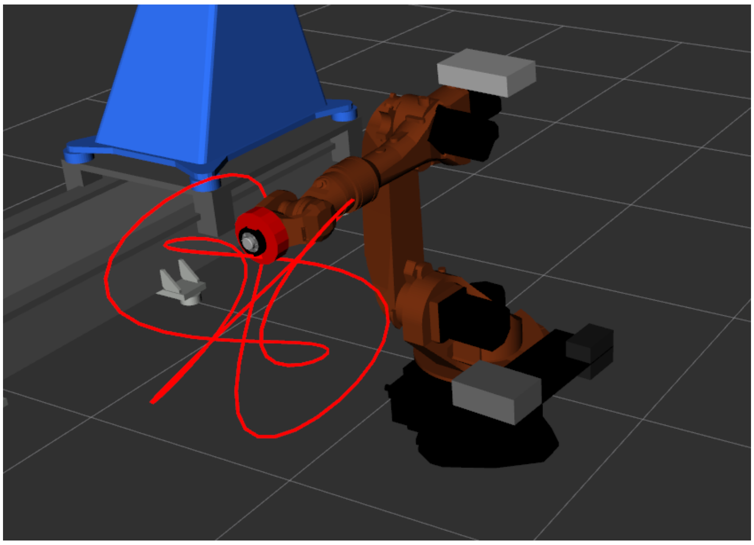


Figure 15: Visualization of the 3D Lissajous trajectory the robot is to follow.

restart a program in ROS without having to restart the program on the KRC, requiring less interaction with the KRC from students.

This layer of abstraction can be problematic for advanced users who want to perform sensor feedback control tasks. To them, RSI is more appropriate. However, the KVP based interface can be used to control tooling and other SPS based aspects of the system while the RSI interface is used for motion. The KVP based interface also allows for logging and plotting the robot motion when executing KRL programs by utilizing the `kvp_joint_state_node`.

Table 2: Comparison of the RSI and KVP based ROS interfaces

| RSI Interface | KVP based Interface |
|---|---|
| Commercial | Open-Source |
| Real-time | Stochastic movement latency |
| Small angle corrections only | Uses KRL motion planner |
| Only stop and start from pendant | Stop, start, and speed from pendant |
| Restart of KRL | No restart of KRL when ROS restarts |
| Ignores workspaces | Uses workspaces |
| Specific KRL for new tools | Read and write to any global booleans |

When it comes to data access and acquisition, it would be more appropriate to compare the KVP based interface with EKI which is meant for data commu-

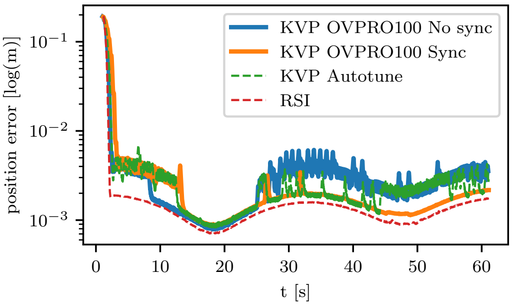


Figure 16: Tracking error of the closed-loop inverse kinematics example.

nication from the KRC to an external computer. This was not done as the core of this investigation was outlining how KVP compares in timing for feedback control education on realistic industrial hardware.

## 5. Conclusion

This article presents a new open-source interface for controlling KUKA robots using ROS, and compares it to a common commercially available interface. It outlines the differing behavior, showing how the commercial interface is more reliable for feedback control tasks, but that with an override speed equal to the required speed of the robot during the task, the open-source interface exhibits similar performance. It also provides an insight into access time latency and movement latency, two factors that are important when designing hardware interfaces.

The two interfaces are not mutually exclusive, and one can use KVP for tooling, statemachines, or observing the joint states while using RSI for motion. The KVP based interface also has some qualities that are helpful in education, e.g. the students do not need to interact with the KRC itself, the educator can control the execution speed from the pendant while the students are using the robot, and it uses the internal trapezoidal motion planner.

Further work includes expanding *BoostCrossCom* to support a wider range of KRL datatypes, allowing for a greater range of tools, sensors, and inspection. Extending the interface with the ability to use Cartesian poses with Euler angles instead of joint poses will make the interface easier to use for non-robotics students and researchers. The degree of flexibility with KVP based interfaces allow for inspecting any global variables in the KUKA robot controller, this can be beneficial in Industry 4.0 as one can create a fully open-source digital twin system using tools in the ROS community such as Gazebo and RViz [23].

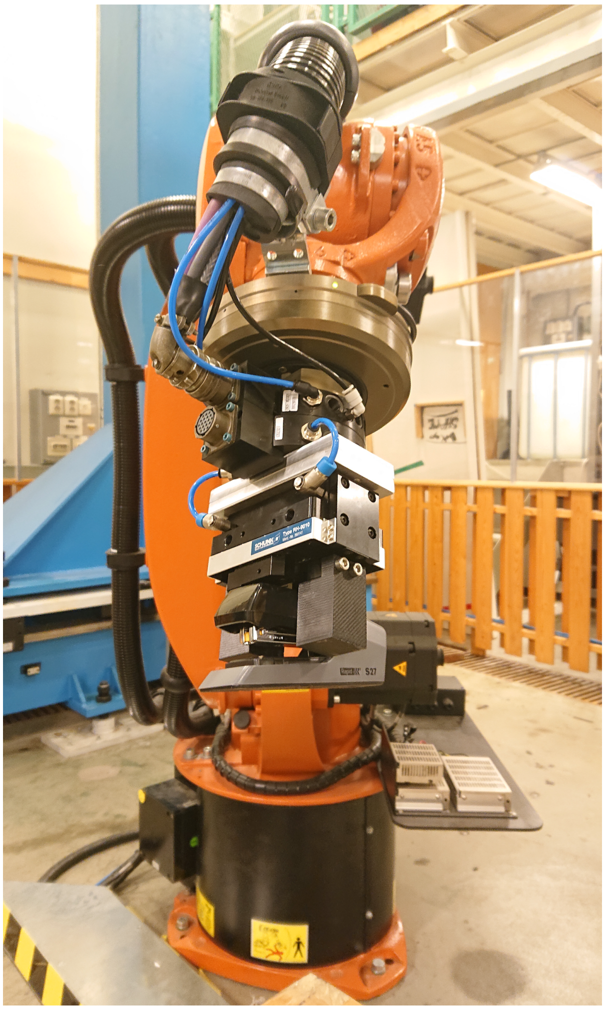


Figure 17: KR 16 ready to transfer the stapler from one location to another using a pneumatic SCHUNK gripper with 3D printed fingers.

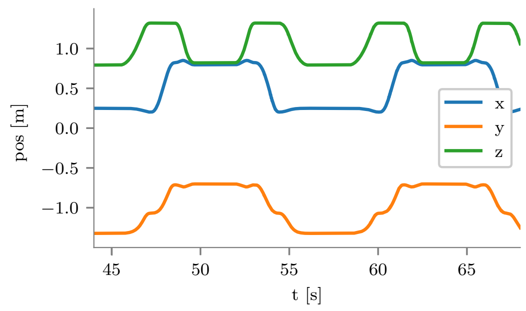


Figure 18: Cartesian coordinates of the end-effector when moving the stapler from one box to another using the KVP interface.

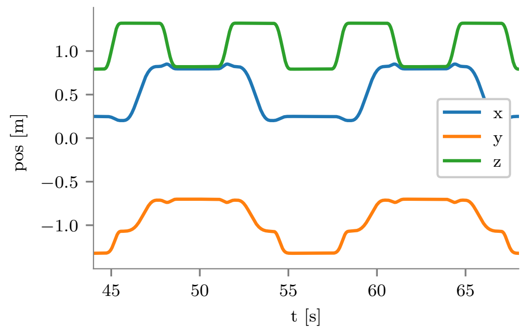


Figure 19: Cartesian coordinates of the end-effector when moving the stapler from one box to another using the RSI interface.

Although this article only consider the KUKA robot platform, the results are important considerations for control theory applications in general. Both the movement latency and the tracking latency are aspects that can negatively impact the transition from academic results to industrial application. Documenting and describing these in a thorough manner are integral for ensuring that one finds and fights the problems where they truly lie.

[1] X. V. Wang, L. Wang, A. Mohammed, M. Givehchi, Ubiquitous manufacturing system based on cloud: A robotics application, Robotics and Computer-Integrated Manufacturing 45 (2017) 116–125.

[2] P. Alhama Blanco, F. Abu-Dakka, M. Abderrahim, Practical use of robot manipulators as intelligent manufacturing systems, Sensors 18 (9) (2018) 2877.

[3] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, A. V. Vasilakos, Software-defined industrial internet of things in the context of industry 4.0, IEEE Sensors Journal 16 (20) (2016) 7373–7380.

[4] H. Mühe, A. Angerer, A. Hoffmann, W. Reif, On reverse-engineering the kuka robot language, arXiv preprint arXiv:1009.5004.

[5] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, K. Y. Pettersen, Jopenshowvar: an open-source cross-platform communication interface to kuka robots, in: Proc. of the IEEE International Conference on Information and Automation (ICIA), Hailar, China, 2014, pp. 1154–1159.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, Ros: an open-source robot operating system, in: Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, 2009.

[7] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, E. Fernández Perdomo, ros_control: A generic and simple control framework for ros, The Journal of Open Source Software-doi:10.21105/joss.00456.

[8] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, K. Y. Pettersen, Controlling kuka industrial robots: Flexible communication interface jopenshowvar, IEEE Robotics Automation Magazine 22 (4) (2015) 96–109. doi:10.1109/MRA.2015.2482839.

[9] E. B. Njåstad, Robotic welding with correction from 3d camera, Master's thesis, Norwegian University of Science and Technology (2015).

[10] I. Eriksen, Setup and interfacing of a kuka robotics lab, Master's thesis, Norwegian University of Science and Technology (2017).

[11] M. Schopfer, F. Schmidt, M. Pardowitz, H. Ritter, Open source real-time control software for the kuka light weight robot, in: Proc. of the 8th IEEE World Congress on Intelligent Control and Automation (WCICA), Jinan, China, 2010, pp. 444–449.

[12] F. Chinello, S. Scheggi, F. Morbidi, D. Prattichizzo, Kuka control toolbox, IEEE Robotics & Automation Magazine 18 (4) (2011) 69–79.

[13] F. Sanfilippo, L. I. Hatledal, H. Zhang, W. Rekdalsbakken, K. Y. Pettersen, A wave simulator and active heave compensation framework for demanding offshore crane operations, in: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), 2015, pp. 1588–1593. doi:10.1109/CCECE.2015.7129518.

[14] S. H. Bredvold, Robotic welding of tubes with correction from 3d vision and force control, Master's thesis, Norwegian University of Science and Technology (2015).

[15] B. Yao, Z. Zhou, L. Wang, W. Xu, Q. Liu, A. Liu, Sensorless and adaptive admittance control of industrial robot in physical human-robot interaction, Robotics and Computer-Integrated Manufacturing 51 (2018) 158 – 168.

[16] S. K. Venugopal, Robot assisted 3d vibrometer measurements with one vibrometer, Master's thesis, Technische Universität Braunschweig (2018).

[17] E. B. Njaastad, O. Egeland, Automatic touch-up of welding paths using 3d vision, IFAC-PapersOnLine 49 (31) (2016) 73 – 78, 12th IFAC Workshop on Intelligent Manufacturing Systems IMS 2016.

[18] I. Eriksen, Itk-thrivaldi github project (December 2017).
URL `https://github.com/itk-thrivaldi/`

[19] I. Eriksen, kuka_kvp_hw_interface (December 2017).
URL `https://github.com/itk-thrivaldi/kuka_kvp_hw_interface`

[20] M. Lind, J. Schrimpf, T. Ulleberg, Open Real-Time Robot Controller Framework, 2010 3rd CIRP Conference on Assembly Technology and Systems (June 2010) (2010) 13–18.

[21] M. H. Arbo, E. I. Grøtli, J. T. Gravdahl, Casclik: Casadi-based closed-loop inverse kinematics, IEEE Transactions on Automation Science and Engineering.

[22] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, CasADi – A software framework for nonlinear optimization and optimal control, Mathematical Programming Computation.

[23] A. Øvern, Industry 4.0 - digital twins and opc ua, Master's thesis, Norwegian University of Science and Technology (2018).

Table A.3: Access time for READ operations in ms for *BoostCrossCom*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 1.72 | 16.10 | 2.37 | 2.06 | 0.83 |
| `INT` | 5 | 2.01 | 30.72 | 10.95 | 10.32 | 1.28 |
| `E6AXIS` | 1 | 1.80 | 52.20 | 2.49 | 2.15 | 0.87 |
| `E6AXIS` | 5 | 2.21 | 32.78 | 11.54 | 11.11 | 1.33 |

Table A.4: Access time for WRITE operations in ms for *BoostCrossCom*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 1.64 | 16.04 | 2.28 | 1.98 | 0.83 |
| `INT` | 5 | 2.00 | 24.33 | 10.64 | 9.94 | 1.30 |
| `E6AXIS` | 1 | 2.54 | 16.85 | 3.42 | 3.07 | 0.95 |
| `E6AXIS` | 5 | 2.95 | 31.03 | 15.10 | 15.66 | 1.37 |

Table A.5: Access time for SYNC operations in ms for *BoostCrossCom*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 6.19 | 35.98 | 23.91 | 23.68 | 1.21 |
| `INT` | 5 | 11.30 | 53.94 | 24.01 | 22.18 | 3.84 |
| `E6AXIS` | 1 | 6.41 | 36.69 | 23.92 | 24.11 | 1.13 |
| `E6AXIS` | 5 | 6.51 | 59.49 | 30.27 | 24.10 | 9.51 |

Table A.6: Access time for READ operations in ms for *JOpenShowVar*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 1.68 | 19.97 | 2.36 | 2.04 | 0.85 |
| `INT` | 5 | 2.01 | 39.56 | 11.02 | 10.34 | 1.35 |
| `E6AXIS` | 1 | 1.75 | 17.90 | 2.42 | 2.09 | 0.84 |
| `E6AXIS` | 5 | 2.08 | 38.12 | 11.33 | 10.67 | 1.42 |

Table A.7: Access time for WRITE operations in ms for *JOpenShowVar*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 1.63 | 20.96 | 2.27 | 1.97 | 0.85 |
| `INT` | 5 | 1.95 | 42.14 | 10.72 | 10.00 | 1.40 |
| `E6AXIS` | 1 | 2.61 | 89.24 | 3.44 | 3.10 | 1.06 |
| `E6AXIS` | 5 | 2.96 | 51.47 | 15.22 | 15.78 | 1.52 |

Table A.8: Access time for SYNC operations in ms for *JOpenShowVar*.

| Data | Threads | Min | Max | Mean | Median | Std.Dev. |
|---|---|---|---|---|---|---|
| `INT` | 1 | 8.07 | 46.35 | 23.93 | 23.67 | 1.20 |
| `INT` | 5 | 8.27 | 64.43 | 30.15 | 31.55 | 5.33 |
| `E6AXIS` | 1 | 8.60 | 64.83 | 23.96 | 24.18 | 1.01 |
| `E6AXIS` | 5 | 8.84 | 80.79 | 34.74 | 33.65 | 9.19 |

# Paper 7 CASCLIK: CasADi-Based Closed-Loop Inverse Kinematics

M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, “CASCLIK: CasADi-Based Closed-Loop Inverse Kinematics,” submitted to IEEE journal of *Transactions on Robotics (T-RO)* January 2019.
Note: The version of the article in this thesis has increased figure sizes and a visualization of the activation map.

# CASCLIK: CasADi-Based Closed-Loop Inverse Kinematics

Mathias Hauan Arbo, *Member, IEEE*, Esten Ingar Grøtli, *Member, IEEE* and Jan Tommy Gravdahl, *Senior Member, IEEE*

***Abstract*—A Python module for rapid prototyping of constraint-based closed-loop inverse kinematics controllers is presented. The module allows for combining multiple tasks that are resolved with a quadratic, nonlinear, or model predictive optimization-based approach, or a set-based task-priority inverse kinematics approach. The optimization-based approaches are described in relation to the set-based task approach, and a novel multidimensional "in tangent cone" function is presented for set-based tasks. A ROS component is provided, and the controllers are tested with matching a pose using either transformation matrices or dual quaternions, trajectory tracking while remaining in a bounded workspace, maximizing manipulability during a tracking task, tracking an input marker's position, and force compliance.**

Note to Practitioners: ***Abstract*—In many industrial use cases, the manipulator is redundant with respect to the task it is to achieve. Placing a symmetrical cup on a table, or spray painting a part rarely requires the full six degrees of freedom to be defined to in order to achieve the task. This redundancy can be exploited to combine multiple tasks such as maximizing manipulability of the task or keeping the shoulder away from an obstacle, and task-based closed-loop inverse kinematic frameworks allow one to do so. As task-based frameworks can involve a significant developmental effort, rapid prototyping of task-based controllers can be used to evaluate whether the alternative technique can result in potential speed up of the task, faster execution, or increased functionality.**

***Index Terms*—Primary Topics: Robot Programming, Motion Control, Secondary Topic Keywords: Force Control**

## I. Introduction

ROBOTS perform tasks that involve interacting and moving objects in Cartesian space by moving joints and motors. Finding control setpoints in terms of the joint coordinates such that the robot can achieve the desired task requires solving the inverse kinematics problem. Inverse kinematics is fundamental to all robots, and occurs in everything from humanoid service robots to 3D printers, surgical robots to autonomous vehicles. In this article we present CASCLIK, a Python module for rapid prototyping of closed-loop inverse kinematics controllers for realizing multiple constraint-based tasks.

Closed-loop inverse kinematics involves defining a feedback controller for achieving the desired task. In [1], Sciavicco et al. present a closed-loop inverse kinematic approach where joint speed setpoints minimize the distance to a given end-effector pose. The distance errors have a guaranteed convergence characteristics. The controller works by inverting the differential kinematics and defines a continuous motion control of the robot.

The task function approach by Samson et al. [2] describes a robotic task as defined by an arbitrary output function and a control objective. The output function is a mapping from the joint states and time to an output space. Samson formulates the task such that the control objective is to bring the output function to zero. The task function approach generalizes to a large class of tasks as the output function may go from any robot states to any positions or orientations defined relative the robot or world frame.

The constraint-based task specification approach of De Schutter et al. [3] describes procedures for designing tasks with complex sensor-based robot systems and geometric uncertainties. Constraint-based task specification uses variables termed *feature variables* to describe position of geometric and task related features that are useful for the task. A key aspect of constraint-based task specification is to allow for feature variables.

A robot is redundant with respect to its task when it has more degrees of freedom than there are dimensions in the output function of the task. This allows one to utilize the free degrees of freedom to achieve tasks simultaneously. A common approach to handling redundancy involves inverting the differential kinematics using a pseudo-inverse. The pseudo-inverse often introduces a null-space within which additional tasks can be achieved. To the author's knowledge, the earliest article combining multiple tasks in this manner is by Hanafusa et al. [4] where a 7 degrees-of-freedom robot tracks a trajectory and avoids an obstacle. This is achieved by placing the lower priority tracking task in the null-space of the higher priority obstacle avoidance task. Chiaverini et al. shows in [5] that multiple tasks can be combined in a singularity robust way. Any framework that supports closed-loop inverse kinematics using task specification should allow for multiple tasks. The state of the art presents two approaches to multiple tasks, strict prioritization with null-space based approaches such as the set-based singularity robust task-priority inverse kinematics framework [6] and optimization-based prioritization which lacks strict priority but allows prioritization through the cost function in an optimization problem [7].

Calculating the Jacobians involved in closed-loop inverse kinematics has been a complicated process requiring explicit

M. H. Arbo and J. T. Gravdahl are with Department of Engineering Cybernetics, NTNU, Norwegian University of Science and Technology.

E. I. Grøtli is with Mathematics and Cybernetics, SINTEF DIGITAL, Trondheim, Norway

The work reported in this paper was supported by the centre for research based innovation SFI Manufacturing in Norway. The work is partially funded by the Research Council of Norway under contract number 237900.

knowledge of the underlying representation used in the tasks. Modern algorithmic differentiation systems such as CasADi [8] simplify this process, allowing us to generate compiled functions of complicated Jacobians. CasADi uses a symbolic framework for performing algorithmic differentiation on expression graphs to construct Jacobians. CasADi provides methods for formulating linear, quadratic, and nonlinear problems that can be solved with e.g. QPOASES [9] and IPOPT [10]. CASCLIK translates a set of tasks to optimal problems of a form that CasADi can solve. This allows the user to test constraint-based programming with any of the available optimizers in CasADi with the different controller formulations presented in this article. The purpose of CASCLIK is to facilitate rapid prototyping of constraint-based control of robot systems.

The architecture of CASCLIK is inspired by eTaSL/eTC [7] by Aertbeliën et al., which is a more mature task specification language and controller. A core principle of the architecture of eTaSL/eTC is to separate the low-level robot controller, numerical solver, and the task specification. Tasks are robot-agnostic and transferrable to any robot system with known forward kinematics. The power of constraint-based task specification and control has allowed the creation of a system architecture capable of exploiting CAD knowledge for assembly [11], for which this work may present alternative controller formulations of interest. Robot-agnostic task specification enables execution of the same task with different robot platforms, which also allows for easier delegation of tasks to the appropriate robots, and transferral of skills from one robot system to another.

CASCLIK is a CasADi-based Python module for testing closed-loop inverse kinematics controllers. The module focuses on being cross-platform and defers to CasADi for the symbolic backend and optimization. The purpose of this module is to explore alternative controller and constraint formulations that utilize the same general structure as eTaSL/eTC. It considers nonlinear and model predictive formulations which are less real-time applicable, in an attempt to investigate aspects that may later be implemented into more industrially relevant frameworks. As it uses CasADi for optimization, CASCLIK utilizes the development efforts of the CasADi community to enable a variety of solvers.

The article is divided into six sections. The first section introduces relevant concepts such as closed-loop inverse kinematics, task function approach, algorithmic differentiation, and presents modern related research. The second section describes the theory involved in CASCLIK. The third section gives a brief description of the implementation. The fourth section gives example applications of CASCLIK and preliminary studies. The fifth and sixth section is the discussion and conclusion.

The main contributions of the article are:

- A nonlinear programming formulation of the constraint-based closed-loop inverse kinematics task controller,
- a model predictive formulation of the constraint-based closed-loop inverse kinematics task controller,
- a general implementation of the set-based singularity robust multiple task-priority inverse kinematics framework of [6],
- a novel multidimensional *in tangent cone* function for the set-based singularity robust multiple task-priority inverse kinematics framework,

## A. Related Research

When considering fundamental robotics problems such as inverse kinematics, there are innumerable important references. To limit the scope we focus on related modern frameworks.

Stack-of-Tasks [12], [13] is a C++ software development kit for real-time motion control of redundant robots. Tasks and robots are defined using *dynamic graphs* that allow for caching results in functions for fast evaluation. The system allows for equality and set tasks by activating and deactivating control of the set tasks. The framework allows for joint torque level control of the robot. Stack-of-tasks also supports a hierarchical quadratic programming formulation [14]. It is open-source, includes tools for integration with ROS, and is limited to Unix platforms.

iTaSC [15], [16] is a software framework for constraint-based task specification and execution. It presents a modular design for task specification, scenegraph representation, and solver. The software framework is a part of the OROCOS project, and uses OROCOS RTT [17], [18] to control robots.

The previously mentioned eTaSL/eTC [7] is a successor to iTaSC, and is a C++/LUA constraint-based task specification and control framework. Expressions are formulated using *expressiongraphs* [19], a symbolic framework that uses OROCOS KDL definitions [20] for frames and rotations. Arbitrary symbolic expressions are used in constraints to form a task specification. The architecture of eTaSL/eTC is modular, allowing one to define new controllers for a task specification and new solvers if they have a C++ interface. It currently supports QPOASES and the hierarchical quadratic programming solver of Stack-of-Tasks. eTaSL/eTC includes a Python interface for rapid prototyping and an OROCOS RTT [17], [18] component for real-time control of robots using OROCOS. eTaSL/eTC is open-source and is currently limited to Linux platforms.

Other advanced constraint-based approaches include the task level robot programming framework of Somani et al. [21], that supports an optimization based solver, and an analytical solver [22]. The software focuses on semantic process description and CAD level tasks and constraints [23]. The CAD level constraints have composition rules, allowing for a reduction of the space of possible control setpoints. The reduced space is used to formulate the analytical solver. To the author's knowledge, the software is not open-source.

The set-based singularity robust multiple task-priority inverse kinematics controller [6] is a task controller that uses the augmented null-space projection operator [24] and activation or deactivation of null-spaces to implement set tasks. This controller forms the null-space approach in CASCLIK and this article extends the approach with support for multidimensional set constraints.

## II. Theory

In this section we present the underlying theory used in CASCLIK. We present the variables and output function involved, the available control objectives one can define, the convexity of the constraints in optimization based controllers, and their effect in the null-space projection based controller. Then we present the four different controllers available: the quadratic, nonlinear, and model predictive optimization-based approaches, and the null-space projection approach. The quadratic programming approach is based on eTaSL/eTC [7] and the null-space approach is based on the set-based singularity robust task-priority inverse kinematics controller [6].

### A. Variables and Output Function

CASCLIK currently supports four different variable types:

- $t$, time,
- $\boldsymbol{q}(t) \in \mathbb{R}^{n_q}$, robot variable (e.g. joint angles),
- $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$, virtual variable (e.g. path-timing),
- $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$, input variables (e.g. sensor values).

Time and robot variables are self-explanatory. Virtual variables are similar to the feature variables of eTaSL/eTC or iTaSC, but the term feature implies a relation to geometric aspects of the task. We describe these as virtual variables as they are variables maintained by the computer, and not necessarily linked to any features of the objects involved. This is merely a semantic choice. Virtual variables simplify task specification and are present in cases such as path-following. Input variables are variables for which we have no information about the derivative behavior.

The output function is a function:

$$\boldsymbol{e}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{n_e}, \tag{1}$$

where $n_e \leq n_q + n_x$ and $t$, $\boldsymbol{x}$ and $\boldsymbol{y}$ are optional. In CASCLIK we assume no knowledge of the underlying geometry involved when evaluating the partial derivatives of the output function. This differs from most other closed-loop inverse kinematics frameworks where the representation is used when evaluating the derivative of transformation matrices and orientations. This is a design choice to make the library as general as possible and allows us to inspect the behavior with different representations. This may require more from the task programmer as the behavior of the robot may differ depending on the formulation of the output function.

**Assumption 1** (Velocity Control)**.** The robot system is equipped with a sufficiently fast velocity controller giving $\dot{\boldsymbol{q}}(t) = \dot{\boldsymbol{q}}_{des}(t)$ where $\dot{\boldsymbol{q}}_{des}$ is the designed control setpoint. The velocity controller controls all robot state velocities.

Samson et al. [2] describe how the first industrial robots had velocity-controlled electrical motors, leading to the joint velocity becoming the *"true control variable"* for robot systems in the control literature. Assumption 1 stems from this time and has been a common robotics assumption since.

### B. Constraints

We use a formulation of robotic tasks similar to Samson et al. [2]: a task is defined by an output function and a control objective. Samson et al. defines the control objective as a regulation problem where a task is performed perfectly during $[t_0, t_f]$ if

$$\boldsymbol{e}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) = 0 \tag{2}$$

for all $t \in [t_0, t_f]$. This is achieved by designing a controller such that the output function converges to zero.

Similar to eTaSL, we refer to the control objective as a type of constraint. CASCLIK specifies four types of constraints:

- equality constraints,

$$\boldsymbol{e}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) = 0, \tag{3}$$

- set constraints,

$$\boldsymbol{e}_l(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{e}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{e}_u(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}), \tag{4}$$

- velocity equality constraints,

$$\dot{\boldsymbol{e}}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) = \dot{\boldsymbol{e}}_d(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}), \tag{5}$$

- and velocity set constraints,

$$\dot{\boldsymbol{e}}_l(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) \leq \dot{\boldsymbol{e}}(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}) \leq \dot{\boldsymbol{e}}_u(t, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}). \tag{6}$$

where subscript $l$ and $u$ refer to the lower and upper bounds, and subscript $d$ refers to a desired derivative of the output function. The control objectives of the tasks are achieved perfectly if the equations hold during $t \in [t_0, t_f]$.

As the control objectives both include equality (converging to zero), and set constraints (converging to or remaining in a set), and set constraints can have different upper and lower bounds, we cannot use the regulation problem formulation of Samson et al. We rely on linearization of the time-derivative of the output functions to achieve the control objectives.

**Assumption 2** (Linearization)**.** The partial derivatives $\frac{\partial \boldsymbol{e}}{\partial t}$, $\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}$ and $\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}$ (commonly called the task Jacobian) can be considered constant with respect to the control duration. That is:

$$\begin{aligned}\frac{\partial \boldsymbol{e}}{\partial t}(\tau) + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}(\tau)\dot{\boldsymbol{q}}(t_n) + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}(\tau)\dot{\boldsymbol{x}}(t_n) \approx \\ \frac{\partial \boldsymbol{e}}{\partial t}(t_n) + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}(t_n)\dot{\boldsymbol{q}}(t_n) + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}(t_n)\dot{\boldsymbol{x}}(t_n)\end{aligned} \tag{7}$$

for $t_n \leq \tau \leq t_n + \Delta_t$ where $t_n$ is a sampling point and $\Delta_t$ is the duration of the control step.

The linearization assumption is often used in closed-loop inverse kinematics frameworks without explicitly stating it as an assumption. The linearization assumption does not always hold, and for long control steps or rapidly moving trajectories a tracking error may occur [25].

Defining a controller for a set of tasks is finding $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{x}})$ such that we achieve the tasks. For the optimization-based controller approaches we do this by imposing constraints on the optimization problem, and for the null-space approach we do this by both null-space projection and inversion of the differential kinematics.

*1) Equality Constraints:* Taking the time-derivative of the output function, we get:

$$\dot{\boldsymbol{e}} = \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \tag{8}$$

where the best guess for the derivatives of $\boldsymbol{y}$ is zero. An equality constraint forms a regulation problem, which is achieved by ensuring that:

$$\dot{\boldsymbol{e}} = \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = -\boldsymbol{K}(t,\boldsymbol{q},\boldsymbol{x},\boldsymbol{y})\boldsymbol{e}(t,\boldsymbol{q},\boldsymbol{x},\boldsymbol{y}) \tag{9}$$

for which exponential convergence to zero is guaranteed if (9) is upheld and $\boldsymbol{K}$ is positive definite. $\boldsymbol{K}$ is a user-defined function, and its dependent variables will be omitted for brevity in the rest of the paper.

Velocity equality constraints are included to allow for velocity following, but do not guarantee convergence:

$$\dot{\boldsymbol{e}} = \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \dot{\boldsymbol{e}}_d(t,\boldsymbol{q},\boldsymbol{x},\boldsymbol{y}), \tag{10}$$

where the right hand side is the desired constraint velocity. This is to accommodate control situations for which the desired output function derivative is easy to define, but its integral is not. Because we rely on Assumption 1 and Assumption 2, the lack of convergence of velocity constraints is not considered in this article.

By using the Moore-Penrose pseudo-inverse (superscript †) we get (9) and (10) on a form that fits with the null-space projection approach. For equality constraints it becomes:

$$\begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{x}}\end{bmatrix} = -\left(\begin{bmatrix}\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\end{bmatrix}\right)^{\dagger}\left(\boldsymbol{K}\boldsymbol{e} + \frac{\partial \boldsymbol{e}}{\partial t}\right). \tag{11}$$

Similarly for the velocity equality constraint we have:

$$\begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\boldsymbol{x}}\end{bmatrix} = \left(\begin{bmatrix}\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}} & \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\end{bmatrix}\right)^{\dagger}\left(\dot{\boldsymbol{e}}_d - \frac{\partial \boldsymbol{e}}{\partial t}\right). \tag{12}$$

*2) Set Constraints:* Set constraints are different in optimization approaches and the null-space projection approach. In optimization based controllers we enable exponential convergence to the set by defining the constraint as:

$$\boldsymbol{K}(\boldsymbol{e} - \boldsymbol{e}_l) \leq \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \leq \boldsymbol{K}(\boldsymbol{e} - \boldsymbol{e}_u) \tag{13}$$

where the gain is defined as previously. Fig.1 visualizes how (13) leads to convergence. When approaching a limit from inside the constraint, the maximum of $\dot{\boldsymbol{e}}$ will gradually be reduced, which causes an exponential decay when approaching a constraint limit.

Similar to the velocity equality constraints, velocity set constraints do not ensure convergence and are defined as:

$$\dot{\boldsymbol{e}}_l(t,\boldsymbol{q},\boldsymbol{x},\boldsymbol{y}) \leq \frac{\partial \boldsymbol{e}}{\partial t} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} \leq \dot{\boldsymbol{e}}_u(t,\boldsymbol{q},\boldsymbol{x},\boldsymbol{y}) \tag{14}$$

which can be visualized as horizontal lines in Fig.1.

Set constraints in the null-space approach are handled using null-space projection and the *in tangent cone* function (Algorithm 1 in [6]). The method states that if the desired $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{x}})$ ensures that the output function remains in the set, by asking whether the $(e, \dot{e})$ pair is in the extended tangent cone, then the set constraint is not active. If the desired robot state velocity is not in the extended tangent cone then the set constraint is active and lower priority tasks are modified by the null-space projection operator of the active set constraint. The choice of $(e, \dot{e})$ pairs that do not cause an activation of the set constraint is visualized in Fig.2.

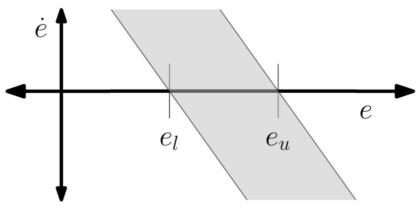


Fig. 1: Visualization of set constraint convergence in one dimension. $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{x}}$ must be chosen such that $\dot{e}$ remains in the gray area. As this results in requiring $\dot{e}$ to be positive when $e < e_l$ and negative when $e > e_u$, we will converge to $e_l \leq e \leq e_u$. The slope of the lines are defined by $K$.

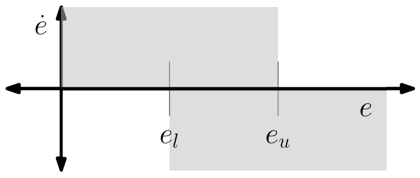


Fig. 2: Visualization of when *in tangent cone* evaluates to true for a one-dimensional output function. For any $(e, \dot{e})$ not in the gray area, the set constraint becomes active and lower level tasks are projected into the null-space of the set constraint.

In [25] it was noted that formulating multidimensional tracking tasks as one dimensional tasks of differing priorities may lead to unexpected tracking errors. The *in tangent cone* function of Moe et al. [6] assumes one-dimensional output functions. CASCLIK addresses this by implementing a multidimensional version which allows for using multidimensional output functions with set constraints. The algorithm is given in Alg.1. If at a time we are at $\boldsymbol{e}$ then the vector $\boldsymbol{d}$ describes the normal vector to the closest point that is in the set. This allows us to identify when $\dot{\boldsymbol{e}}$ points inwards. The signs of $\boldsymbol{e} - \boldsymbol{e}_l$ and $\boldsymbol{e} - \boldsymbol{e}_u$ are equal when the closest point inside the set is on the corners of a set, allowing us to identify the corners as special cases.

The multidimensional *in tangent cone* function assumes that corners can be approximated with a $45^\circ$ cone situated at the corner. This is an approximation that may falsely report that we are not in the extended tangent cone for $\dim(\boldsymbol{e}) > 2$, e.g. when the desired $\dot{\boldsymbol{e}}$ points along an edge of the set constraint.

Velocity set constraints are not currently defined in the task-priority inverse kinematics framework, and are therefore not included in the null-space projection approach.

*3) Convexity of Desired Control Input Space:* For the optimization-based approaches, the task constraints form constraints in the optimization problem. The derivative of the output function is an affine function with respect to the desired

**Algorithm 1** Multidimensional *in tangent cone*

```
Input: $t, q, x, y, \dot{q}, \dot{x}$
1:  $d \leftarrow \text{sign}(e - e_l) + \text{sign}(e - e_u)$
2:  in_crnr $\leftarrow \text{sign}(e - e_l) == \text{sign}(e - e_u)$
3:  if $e_l(t, q, x, y) \leq e(t, q, x, y) \leq e_u(t, q, x, y)$ then
4:      return True
5:  else if in_crnr and $|-d \cdot \dot{e}| < \|d\| \|\dot{e}\| \cos(45°)$ then
6:      return True
7:  else if not in_crnr and $d \cdot \dot{e} < 0$ then
8:      return True
9:  else
10:     return False
11: end if
```

control input $\dot{q}$ and $\dot{x}$. For the different constraint types, the space of desired control inputs are:

$$\mathcal{D}(e, eq) = \left\{ \dot{q}, \dot{x} \middle| \frac{\partial e}{\partial t} + \frac{\partial e}{\partial q}\dot{q} + \frac{\partial e}{\partial x}\dot{x} = -Ke \right\} \tag{15}$$

$$\mathcal{D}(e, vel.eq) = \left\{ \dot{q}, \dot{x} \middle| \frac{\partial e}{\partial t} + \frac{\partial e}{\partial q}\dot{q} + \frac{\partial e}{\partial x}\dot{x} = \dot{e}_d \right\} \tag{16}$$

$$\mathcal{D}(e, set) = \left\{ \dot{q}, \dot{x} \middle| \frac{\partial e}{\partial t} + \frac{\partial e}{\partial q}\dot{q} + \frac{\partial e}{\partial x}\dot{x} \in [K(e - e_l), K(e - e_u)] \right\} \tag{17}$$

$$\mathcal{D}(e, vel.set) = \left\{ \dot{q}, \dot{x} \middle| \frac{\partial e}{\partial t} + \frac{\partial e}{\partial q}\dot{q} + \frac{\partial e}{\partial x}\dot{x} \in [\dot{e}_l, \dot{e}_u] \right\}. \tag{18}$$

At any particular time instance $t, q, x$ and $y$ are constant, making $\dot{e}$ an affine transformation with respect to $\dot{q}$ and $\dot{x}$. From [26] we know that the inverse image of an affine function on a convex set is convex, which makes $\mathcal{D}$ convex. The set of possible choices of $(\dot{q}, \dot{x})$ with multiple tasks is

$$\mathcal{S}(t, q, x, y) = \bigcap_{i=1}^{n_c} \mathcal{D}(e_i, c_i)(t, q, x, y), \tag{19}$$

where we have $n_c$ tasks, each with an output function $e_i$ and a control objective $c_i \in \{eq., vel.eq., set, vel.set\}$. As the intersection of convex sets is convex, combining tasks maintains convexity of the set of possible control variables. This convexity hinges on the derivative of the output function being affine with respect to the control variables. Thus the convexity argument does not hold for the model predictive approach where $q$ and $x$ are predicted variables for the predicted constraints.

Tasks are incompatible if $\mathcal{S} = \emptyset$ (e.g. first task is to remain in a box, second task is to track a reference that leaves the box). We can easily see that adding a slack variable term $\epsilon$ to $\dot{e}$ reinstates the convexity with respect to the variables $(\dot{q}, \dot{x}, \epsilon)$ for the non-predictive approaches.

*4) Null-Space Projection:* Given an output function $e_i$, we define the null-space projection operator of the task as:

$$N_i = I_{n_q+n_x} - \left[\frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x}\right]^\dagger \left[\frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x}\right] \tag{20}$$

such that $N_i v = 0$ if $v$ is a vector that extends only into the space of the task. For multiple tasks, the null-space of all the tasks combined uses the augmented inverse-based projection of [24]:

$$N_{i,i+1,\dots} = I_{n_q+n_x} - \begin{bmatrix} \frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x} \\ \frac{\partial e_{i+1}}{\partial q}, \frac{\partial e_{i+1}}{\partial x} \\ \vdots \end{bmatrix}^\dagger \begin{bmatrix} \frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x} \\ \frac{\partial e_{i+1}}{\partial q}, \frac{\partial e_{i+1}}{\partial x} \\ \vdots \end{bmatrix} \tag{21}$$

With multidimensional set constraints the null-space should only consider directions in which the output function violates the set constraints. For each set constraint we define a diagonal activation matrix $S_i \in \mathbb{R}^{n_{e_i} \times n_{e_i}}$ with diagonal elements:

$$s_{i,j} = \begin{cases} 1, & e_{i,j} < e_{l,i,j} \text{ or } e_{i,j} > e_{u,i,j} \\ 0, & \text{else} \end{cases} \tag{22}$$

where subscript $i, j$ refers to $j$th element of the $i$th output function, upper bound, or lower bound. With the activation matrix, the augmented inverse-based projection becomes:

$$N_{i,i+1,\dots} = I_{n_q+n_x} - \begin{bmatrix} \frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x} \\ \frac{\partial e_{i+1}}{\partial q}, \frac{\partial e_{i+1}}{\partial x} \\ \vdots \end{bmatrix}^\dagger \begin{bmatrix} J_{A,i} \\ J_{A,i+1} \\ \vdots \end{bmatrix} \tag{23}$$

where

$$J_{A,i} = \begin{cases} [\frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x}], & \text{if equality constraint} \\ S_i[\frac{\partial e_i}{\partial q}, \frac{\partial e_i}{\partial x}], & \text{if set constraint.} \end{cases} \tag{24}$$

## C. Quadratic Programming Approach

The quadratic programming (QP) approach is a reactive control that formulates a QP problem based on the current sensor information. At time $t = t_k$, we know $q(t_k)$,$x(t_k)$, and $y(t)$. $\dot{q}_k$ is a setpoint sent to the robot system and $\dot{x}_k$ is used to obtain $x_{k+1}$. The optimization problem is:

$$\min_{\dot{q}_k, \dot{x}_k, \epsilon} \quad c\dot{q}_k^T W_{\dot{q}} \dot{q}_k + c\dot{x}_k^T W_{\dot{x}} \dot{x}_k + (1+c)\epsilon^T W_\epsilon \epsilon \tag{25a}$$

$$s.t.:$$

$$(\dot{q}_k, \dot{x}_k) \in \mathcal{S}(t_k, q(t_k), x(t_k), y(t_k), \epsilon) \tag{25b}$$

where $\mathcal{S}$ is the set of all $(\dot{q}_k, \dot{x}_k)$ such that:

$$\frac{\partial e_i}{\partial t} + \frac{\partial e_i}{\partial q}\dot{q}_k + \frac{\partial e_i}{\partial x}\dot{x}_k = -K_i e_i + \epsilon_i \tag{26}$$

$$K_j(e_j - e_{l,j}) \leq \frac{\partial e_j}{\partial t} + \frac{\partial e_j}{\partial q}\dot{q}_k + \frac{\partial e_j}{\partial x}\dot{x}_k + \epsilon_j \tag{27}$$

$$\frac{\partial e_j}{\partial t} + \frac{\partial e_j}{\partial q}\dot{q}_k + \frac{\partial e_j}{\partial x}\dot{x}_k + \epsilon_j \leq K_j(e_j - e_{u,j}) \tag{28}$$

$$\frac{\partial e_m}{\partial t} + \frac{\partial e_m}{\partial q}\dot{q}_k + \frac{\partial e_m}{\partial x}\dot{x}_k = -\dot{e}_{d,m} + \epsilon_m \tag{29}$$

$$\dot{e}_{l,n} \leq \frac{\partial e_n}{\partial t} + \frac{\partial e_n}{\partial q}\dot{q}_k + \frac{\partial e_n}{\partial x}\dot{x}_k + \epsilon_n \leq \dot{e}_{u,n} \tag{30}$$

where $i \in [0, I]$ are all equality constraints, $j \in [0, J]$ are all set constraints, $m \in [0, M]$ are all velocity equality constraints and $n \in [0, N]$ are all the velocity set constraints. $\epsilon$ denotes a vector of all slack variables, and $c$ is a regularization weight. The matrices $W_{\dot{q}}, W_{\dot{x}}$ and $W_\epsilon$ are positive definite matrices denoting the weights on the robot velocity, virtual variable velocity, and the slack variables respectively.

This formulation is based on the QP controller in eTaSL/eTC. The gains, output functions, and partial derivatives of the output functions are evaluated at time $t = t_k$ and assumed constant with respect to the optimization problem. When the controller is started for the first time, the virtual variables and the slack variables must be initialized. This is done by solving the QP problem (25) at time $t = t_0$ with $\dot{\boldsymbol{q}}_0 = 0$.

The slack variable defines the behavior when constraints are incompatible. This is a form of soft "prioritization" of the constraints by avoiding the case where $\mathcal{S} = \emptyset$. With the QP approach all objectives of the controller are formulated in terms of constraints.

### D. Nonlinear Programming Approach

The nonlinear programming (NLP) approach is a reactive control approach that uses the same problem formulation as the QP approach, but allows for more general cost expressions. The optimization problem is:

$$\min_{\dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k, \boldsymbol{\epsilon}} \quad cf(t_k, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}, \dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k) + (1+c)\boldsymbol{\epsilon}^T W_{\boldsymbol{\epsilon}} \boldsymbol{\epsilon} \tag{31a}$$

$$s.t. :$$

$$(\dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k) \in \mathcal{S}(t_k, \boldsymbol{q}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\epsilon}_k) \tag{31b}$$

where the cost function $f$ is a user-defined function. If the cost function is convex, then we have a convex optimization problem for which efficient solvers exist. The cost function must depend upon $\dot{\boldsymbol{q}}_k$, and $\dot{\boldsymbol{x}}_k$ if there are virtual variables, the rest are optional.

As the QP and NLP approach are similar in their constraints and regularization, any set of tasks implemented for the QP approach can be implemented for the NLP approach. The NLP approach allows defining more complex controllers by implementing objectives in terms of costs.

### E. Model Predictive Approach

The QP and NLP approaches are reactive approaches where the current states of the robot system are used to determine what the next control input should be. A natural extension to such a system is the introduction of model predictive control (MPC). The states are predicted by relying on Assumption 1. The MPC approach does not support inputs $\boldsymbol{y}$ as there is no way of predicting what the input will be.

The MPC approach problem is implemented using a multiple-shooting strategy. We define the horizon length as $n_h$ steps of length $\Delta_t$ and the optimization variable $\boldsymbol{\chi} = \{\dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k, \epsilon_k, \boldsymbol{q}_{k+1}, \boldsymbol{x}_{k+1}, \}_{k\in[0,n_h-1]}$. The times are $t_k = t_0 + \Delta_t k$. The control input duration is $\Delta_t$. The problem is formulated as:

$$\min_{\boldsymbol{\chi}} \quad \Phi(\boldsymbol{\chi}) \tag{32a}$$

$$s.t. :$$

$$\boldsymbol{q}_0 = \boldsymbol{q}(t_0) \tag{32b}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}(t_0) \tag{32c}$$

$$\boldsymbol{q}_{k+1} - (\boldsymbol{q}_k + \dot{\boldsymbol{q}}_k \Delta_t) = 0 \tag{32d}$$

$$\boldsymbol{x}_{k+1} - (\boldsymbol{x}_k + \dot{\boldsymbol{x}}_k \Delta_t) = 0 \tag{32e}$$

$$(\dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k) \in \mathcal{S}(t_k, \boldsymbol{q}_k, \boldsymbol{x}_k) \tag{32f}$$

where $k \in [0, n_h]$ and

$$\Phi(\boldsymbol{\chi}) = \sum_{k=0}^{n_h-1} cf(t_k, \boldsymbol{q}_k, \boldsymbol{x}_k, \dot{\boldsymbol{q}}_k, \dot{\boldsymbol{x}}_k) + (1+c)\boldsymbol{\epsilon}_k^T \boldsymbol{W}_{\boldsymbol{\epsilon}} \boldsymbol{\epsilon}_k \tag{33}$$

is the cost from the NLP applied to each timestep along the prediction horizon. (32b)-(32c) are lifting conditions for the current timestep. (32d)-(32e) are the shooting gap constraints. $\mathcal{S}$ uses either the shooting-gap variables for the predicted constraints or the numerical value for the initial constraints. As mentioned, the convexity of the task constraints is not guaranteed for the shooting-gap variables. This is because the terms depend on predictions rather than constants, and the derivative of the output function is not necessarily affine.

The MPC approach is a bridge between closed-loop inverse kinematics and motion planning. The MPC approach may utilize knowledge along its prediction horizon to choose more appropriate control inputs. This comes at the cost of computational complexity, and not guaranteeing convexity of the constraints. With $n_q$ robot variables, $n_x$ virtual variables, $n_\epsilon$ slack variables, and the dimension of the task constraints as $n_c$, the number of decision variables for the QP and NLP approach is $n_q + n_x + n_\epsilon$ and there are $n_c$ constraint equations. For the MPC approach the number of decision variables is $n_h(2n_q + 2n_x + n_\epsilon)$ and the dimension of the constraints is $n_h(2n_q + 2n_x + n_c)$.

### F. Null-Space Projection Approach

As previously stated, the null-space projection approach comes from the singularity robust task-priority framework of Moe et al. [6]. Tasks have a strict prioritization as lower priority tasks are projected into the null-space of higher priority tasks.

Given a priority sorted sequence $i \in [0, \dots, n_c - 1]$ of constraints, the desired control variables are:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{x}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}}_{d,0} \\ \dot{\boldsymbol{x}}_{d,0} \end{bmatrix} + \sum_{j=1}^{n_c-1} \boldsymbol{N}_{0,\dots,j} \begin{bmatrix} \dot{\boldsymbol{q}}_{d,j} \\ \dot{\boldsymbol{x}}_{d,j} \end{bmatrix} \tag{34}$$

where $[\dot{\boldsymbol{q}}_{d,0}^T, \dot{\boldsymbol{x}}_{d,0}^T]^T$ and $[\dot{\boldsymbol{q}}_{d,j}^T, \dot{\boldsymbol{x}}_{d,j}^T]^T$ are defined by (11) for equality constraints, (12) for velocity equality constraints, and $\boldsymbol{0}$ for set constraints. If a set constraint is inactive, it does not contribute to the augmented null-space projection of its lower priority tasks. If it is active, its task Jacobian is used when formulating the null-space projector. The null-space is formed using (21) or (23) when using the multidimensional *in* *tangent cone* function. With $n_{set}$ set based tasks, we have $2^{n_{set}}$ possible activation combinations of the set constraints. These form $2^{n_{set}}$ possible modes of the controller. In Table I we see the activation of map for an example with 3 set constraints, thus having 8 modes. Check marks are active, dash marks are inactive set constraints. At each timestep, we check each

TABLE I: Activation map with 3 set constraints

| Mode | Set 1 | Set 2 | Set 3 |
|---|---|---|---|
| 1 | - | - | - |
| 2 | - | - | ✓ |
| 3 | - | ✓ | - |
| 4 | ✓ | - | - |
| 5 | - | ✓ | ✓ |
| 6 | ✓ | - | ✓ |
| 7 | ✓ | ✓ | - |
| 8 | ✓ | ✓ | ✓ |

mode for whether the $(\dot{\boldsymbol{q}}, \dot{\boldsymbol{x}})$ it proposes is in the extended tangent cone for all inactive set constraints. This is done by going down the list of modes, activating set constraints until all other set constraints are in their extended tangent cones. At worst we will evaluate each mode and the times *in tangent cone* is run is $O(n_{set}\log_2(n_{set}))$ [27].

The null-space projection approach gives hard limits on the constraints. It ensures that we cannot chose control inputs that go out of the sets. This differs from the optimization approaches where the control variables are chosen such that the controller converges to the sets. If external disturbances, numerical errors, or measurement noises causes the null-space projection controller to end up outside the edge of a set constraint, the controller will still attempt to move along the null-space of the constraint, and not necessarily converge into the set again. This means that one must start the controller with the system inside the sets.

## III. Implementation

In this section we briefly present the implementation of CASCLIK and the support packages. This is to give insight into their purpose, and important design choices.

### A. CasADi - Jacobian Damping

As CasADi is a symbolic framework, it performs pseudo-inverse by assuming that the item to be inverted has full rank. If $\boldsymbol{M}$ is the matrix to be inverted CasADi solves the linear problem:

$$\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{x} = \boldsymbol{M} \tag{35}$$

for $\boldsymbol{x}$ if $\boldsymbol{M}$ is wide, or by

$$\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{x} = \boldsymbol{M} \tag{36}$$

if $\boldsymbol{M}$ is tall. We employ Jacobian damping [28] to give the Jacobian full rank as the activation matrix $\boldsymbol{S}_i$ in the multidimensional set constraint may lead to zero rows in $\boldsymbol{J}_A$, or ill-defined tasks may lead to zero rows. This modifies (35) to solve:

$$(\boldsymbol{M}\boldsymbol{M}^T + \lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{M} \tag{37}$$

for wide matrices, where $\lambda$ is the damping factor, and similar for tall matrices. The default damping factor is set to $10^{-7}$ and is an option in the null-space approach controller.

### B. CASCLIK

CASCLIK is a Python module that only depends on CasADi. This is to have an operating system independent, robot middleware independent software solution. The module is compatible with both Python 2.7 and Python 3. CASCLIK is available on GitHub under the MIT license [29]. The overall architecture is inspired by eTaSL/eTC. The core module contains classes for constraints, skill specification, and controllers.

The output function of a constraint is an arbitrary CasADi expression. The gain, target derivative, and upper or lower limits can be added depending on what type of control objective is involved. Priority is added by specifying the constraint as soft or hard for the optimization-based approaches, or as a numerical value for the null-space based approach.

A collection of constraints is a skill. As the user is free to define both what the time, robot, virtual, and input variables are called when formulating the constraints, the user must provide the symbol for each of the relevant variables to the skill specification as well as a label and a list of constraints. The skill sorts the constraints according to their numerical priority (relevant for the null-space projection approach), and keeps track of whether there are slack variables or virtual variables involved in the skill.

Controllers take a skill specification and other controller-dependent parameters as well as an option dictionary. The *ReactiveQPController* (QP) takes a list of weights for the robot or virtual variables. The *ReactiveNLPController* (NLP) takes a cost expression. The *ModelPredictiveController* (MPC) takes a cost expression as well as the horizon length, and a timestep length. The null-space based *PseudoInverseController* (PINV) has no optional input.

The controllers are compiled using CasADi's just-in-time compilation of solvers and functions. For problems containing a large number of sets, PseudoInverseController has generally the longest compilation time as there are $2^{n_{set}}$ separate modes to compile.

### C. Other modules

Two additional modules were created to simplify prototyping. *urdf2casadi* is a Python module for generating CasADi expressions for the forward kinematics of robots. It uses either URDF files, which are common in ROS, or Denavit-Hartenberg parameters, common in industry, for creating forward kinematics reprented by a transformation matrix or a dual quaternion. *casclik_basics* provides classes for interfacing with robots that maintain the virtual variables and subscribes to joint and sensor topics. Its *DefaultRobotInterface* publishes joint position commands, and its *URModernInterface* is specifically intended for use with the *ur_modern_driver* [30] and publishes joint speed commands. *casclik_basics* is available on GitHub under the MIT License [29].

## IV. Examples

The tests were performed on a computer with an Intel Xeon CPU E5-1650 v3 running Ubuntu 16.04 with ROS Kinetic Kame. In all the experiments we use QPOASES for the QP controller, IPOPT for the NLP and MPC controller, and Jacobian damping for the PINV controller (null-space approach).

### A. Representation - Matrix or Quaternion

In this example we are controlling a UR5 robot using either dual quaternions or transformation matrices for frame representation. The example uses the UR5 URDF with *urdf2casadi* to determine forward kinematics. The robot is simulated at joint velocity level with Euler discretization as we rely on Assumption 1.

Transformation matrices $\boldsymbol{T} \in \mathcal{T} \subset \mathbb{R}^{4\times4}$ are composed of a rotation matrix $\boldsymbol{R} \in \mathcal{R} \subset \mathbb{R}^{3\times3}$, and a displacement vector $\boldsymbol{p} \in \mathbb{R}^3$. Dual quaternions $\breve{Q}$ are composed of

$$\breve{Q} = Q_R + \varepsilon Q_p \tag{38}$$

where $Q_R \in \mathcal{Q}_{unit}$ is a unit quaternion for rotation and $Q_p \in \mathcal{Q}$ a quaternion for displacement. $\varepsilon$ is the dual unit which satisfies $\varepsilon\varepsilon = 0$. Dual quaternions can be represented by a vector such that $\breve{\boldsymbol{Q}} \in \mathbb{R}^8$, and in vector form the quaternion product of two dual quaternions $\breve{Q}_c = \breve{Q}_a \otimes \breve{Q}_b$ can be defined as:

$$\breve{\boldsymbol{Q}}_c = \bar{\boldsymbol{H}}(\breve{\boldsymbol{Q}}_b)\breve{\boldsymbol{Q}}_a = \boldsymbol{H}(\breve{\boldsymbol{Q}}_a)\breve{\boldsymbol{Q}}_b \tag{39}$$

where $\bar{\boldsymbol{H}}, \boldsymbol{H} \in \mathbb{R}^{8\times8}$ are matrices referred to as the minus and plus Hamilton operator [31].

The UR5 has $\boldsymbol{q} \in \mathbb{R}^6$ joint angles forming the robot variables, and forward kinematics described by $\boldsymbol{R}(\boldsymbol{q})$ for the rotation matrix, $\boldsymbol{p}(\boldsymbol{q})$ for the displacement, and $\breve{\boldsymbol{Q}}(\boldsymbol{q})$ for the dual quaternion.

The task is for the end-effector frame to match a desired frame. The desired frame is described by $(\boldsymbol{R}_d, \boldsymbol{p}_d)$ with transformation matrices, and $\breve{\boldsymbol{Q}}_d$ with dual quaternions. Using rotation and displacement we can define this as the task:

$$\boldsymbol{e}_T(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{p}(\boldsymbol{q}) - \boldsymbol{p}_d \\ \left\| \boldsymbol{R}_d^T \boldsymbol{R}(\boldsymbol{q}) - \boldsymbol{I} \right\|_F \end{bmatrix} \tag{40}$$

where the first line ensures convergence of position and the second ensures convergence of the rotation. The second line uses the orientation metric of Larochelle et al. [32] using the Frobenius norm.

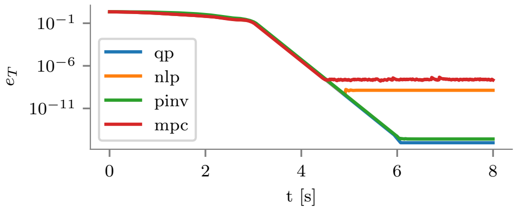
For dual quaternions, we employ the strategy of Figueredo et al. [31]:

$$\boldsymbol{e}_Q(\boldsymbol{q}) = \bar{\boldsymbol{H}}(\breve{\boldsymbol{Q}}_d)\boldsymbol{C}(\breve{\boldsymbol{Q}}_d - \breve{\boldsymbol{Q}}(\boldsymbol{q})) \tag{41}$$

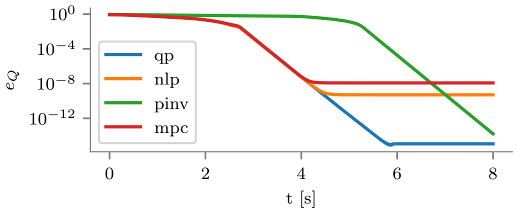
where $\boldsymbol{C} = \mathrm{diag}(-1,-1,-1,1,-1,-1,-1,1)$ is the conjugate operator for dual quaternions in vector form. As $\boldsymbol{Q}_d$ is constant, we see that (41) becomes linear with respect to $\breve{\boldsymbol{Q}}(\boldsymbol{q})$.

The joints have hard set constraints such that $q_i \in [-2\pi, 2\pi]$, and $\dot{q}_i \in [-\pi/5, \pi/5]$. As the null-space based controller does not support velocity set constraints, the applied $\dot{\boldsymbol{q}}$ is also saturated by the max speed. The example is simulated with a desired frame at $\boldsymbol{p}_{des} = [0.5, 0, 0.5]^T$, with a roll of 5°. The control duration is 8 ms, and corresponds to 125 Hz. The MPC approach has a prediction horizon of 10 control steps. All cost functions are the same as for the QP approach.

In Fig.3 we see the Euclidean norm of the two representations for each of the controller classes. The null-space approach has a greater error while moving closer to the point



(a) $\boldsymbol{e}_T$



(b) $\boldsymbol{e}_Q$

Fig. 3: Euclidean norm of the output functions (40) and (41).

as it does not account for the speed saturation. It also struggles more with the dual quaternion formulation and takes a more circuitous route. The different controllers have different limits before numerical issues arise and these may be optimizer settings dependent.

In Tab.II the initial and average runtimes are given for the different controllers during the simulations. The null-space approach is denoted by PINV.

TABLE II: Controller runtimes for Representation Example

| | PINV | QP | NLP | MPC |
|---|---|---|---|---|
| Initial ($\boldsymbol{e}_Q$) | 0.11 ms | 0.95 ms | 4.23 ms | 26.80 ms |
| Average ($\boldsymbol{e}_Q$) | 0.04 ms | 0.27 ms | 2.74 ms | 20.01 ms |
| Initial ($\boldsymbol{e}_T$) | 0.09 ms | 1.44 ms | 4.37 ms | 16.90 ms |
| Average ($\boldsymbol{e}_T$) | 0.04 ms | 0.26 ms | 2.81 ms | 147.08 ms |

The NLP approach uses approximately half the control duration, and MPC approach generally uses an order of magnitude longer. This means that the controllers would not be applicable to this control situation with the default settings. The QP approach and null-space approach have applicable timings with the QP approach being an order of magnitude slower than the null-space approach. The average runtime of the MPC formulation with transformation matrix formulation is much higher as a result of using the Frobenius norm and matrix operations. This is likely caused by the prediction constraints becoming more difficult to apply.

### B. Set Constraints - Bounded Workspace

This is a recreation of Example 2 from [6] where a UR5 tracks a Cartesian trajectory while not escaping a box defined workspace. The forward kinematics are defined by the Denavit-Hartenberg parameters in [6] using *urdf2casadi*.

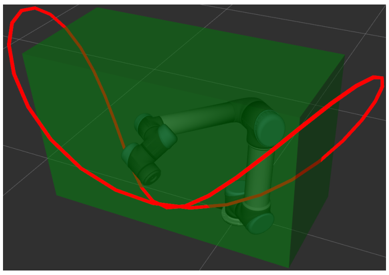


Fig. 4: RViz visualization of the bounded workspace example.

Listing 1: Equality Constraint Example

```
track_cnstr = EqualityConstraint(
   label="tracking_constraint",
   expression=p(q) - p_d(t),
   gain=1.0,
   constraint_type="soft",
   priority=3
)
```

The output function is defined by

$$\boldsymbol{e}(t,\boldsymbol{q}) = \boldsymbol{p}(\boldsymbol{q}) - \boldsymbol{p}_{des}(t) \tag{42}$$

where $\boldsymbol{p}(\boldsymbol{q})$ is the forward kinematics to the origin of the end-effector, and

$$\boldsymbol{p}_{des}(t) = \begin{bmatrix} 0.5\sin^2(0.1t)+0.2 \\ 0.5\cos(0.1t)+0.25\sin(0.1t) \\ 0.5\sin(0.1t)\cos(0.1t)+0.7 \end{bmatrix}. \tag{43}$$

The set constraint is defined by $\boldsymbol{p}(\boldsymbol{q}) \in [\boldsymbol{p}_l, \boldsymbol{p}_u]$ with $\boldsymbol{p}_l = [0.1, -0.5, 0.3]$ and $\boldsymbol{p}_l = [0.5, 0.4, 0.85]$. Examples of the code used to define the equality and set constraints can be seen in Listing.1 and Listing.2. For the null-space projection approach the set constraint can either be formulated using the experimental multidimensional set constraint, or as three separate constraints for $x$, $y$, and $z$ as in [6]. The equality constraint has a lower priority (3rd) such that it can work with either formulation.

From Fig.1 we know that the approach speed to the upper or lower bound on a set constraint are determined by the gain in the optimization approaches. This can be seen as an exponential decay in the tracking task as we approach the set limits. From Fig.2 we see that as the gain approaches $\infty$, we will have the same sharp change when approaching a set limit as the null-space approach exhibits. In this example the set gain is 100 for the QP and NLP. The MPC has gains of 1 as large set gains may lead to more difficult predicted constraints.

Listing 2: Set Constraint Example

```
box_cnstr = SetConstraint(
   label="box_constraint",
   expression=p(q),
   set_min = np.array([0.1, -0.5, 0.3]),
   set_max = np.array([0.5, 0.4, 0.85]),
   gain = 100.0,
   constraint_type="hard",
   priority=1
)
```

In Fig.5 we see the position of the end-effector for the different controllers when handling $x$, $y$, and $z$ as separate constraints and in Fig.6 we see the position when handling them as a single multidimensional constraint.

In Fig.7 we see the tracking error with the different controllers when handling $x$, $y$, and $z$ constraints as separate constraints. In Fig.8 we see the tracking error with the different controllers when handling $x$, $y$, and $z$ as a single multidimensional constraint. Similar to the results in [25], setting the constraints in a priority ordered sequence causes unwanted behavior. The multidimensional formulation gives a more correct interpretation of the constraint. Also note that the multidimensional null-space approach and the optimization approaches are similar. If more set constraints are included, such as joint limits, the two methods will differ again. In Fig.9 we see the mode the null-space based controller is in for both the separate $x$, $y$, and $z$ formulation and the multidimensional formulation. The "noisy" rapid switching of modes occurs due to numerical issues with the linearization and the comparison between current and set limits. Tuning either the control duration or the comparison with some numerical lower limit can mitigate this effect.

Fig.4 shows a visualization of the controller running with the DefaultRobotInterface and ROS.

### C. Nonlinear cost - Manipulability Index

In this example a 7 degrees of freedom KUKA LWR IIWA 14 R820 arm is to follow a circular trajectory in its workspace and maximize the manipulability index of the task. The forward kinematics are defined by the URDF and *urdf2casadi*. The robot is simulated at joint velocity level with Euler discretization. IIWA has $\boldsymbol{q} \in \mathbb{R}^7$ where $\boldsymbol{q} \in [-\boldsymbol{q}_u, \boldsymbol{q}_u]$ with

$$\boldsymbol{q}_u^T = [170°, 120°, 170°, 120°, 170°, 120°, 175°]. \tag{44}$$

The circular trajectory is defined by:

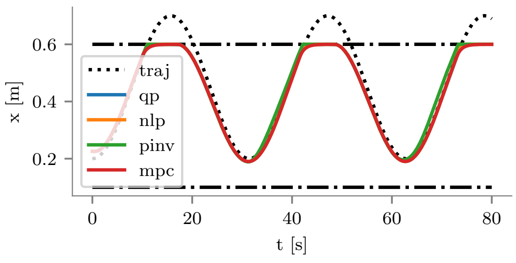
$$\boldsymbol{p}_d(t) = \begin{bmatrix} 0.1\cos(0.05t-0.5\pi)+0.45 \\ 0.1\sin(0.05t-0.5\pi)+0.4 \\ 0.3 \end{bmatrix}. \tag{45}$$

We define the manipulability index of the task as

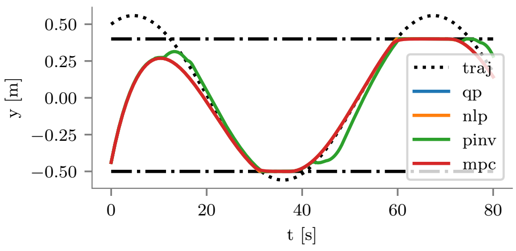
$$m(\boldsymbol{q}) = \sqrt{\det\left(\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{q}}(\boldsymbol{q})\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{q}}(\boldsymbol{q})^T\right)} \tag{46}$$

where $m$ is a measure of the area of the ellipsoid that the Jacobian of the end-effector position forms. We want to both achieve the task and to maximize our manipulability. Maximizing the manipulability can be beneficial in collision avoidance, as a high manipulability means we have more options as to which direction we can move to avoid collision. Maximizing the manipulability can be achieved by adding a term to the nonlinear costs of the NLP and the MPC approach:

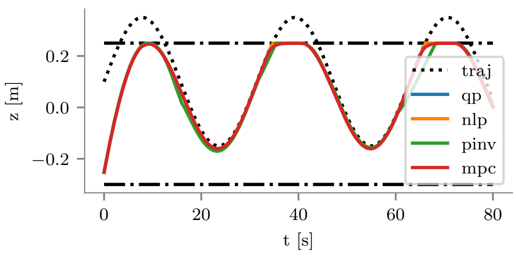
$$m_c(\boldsymbol{q},\dot{\boldsymbol{q}}) = -\alpha m(\boldsymbol{q}+\Delta_t\dot{\boldsymbol{q}})^2 \tag{47}$$



(a) $x$

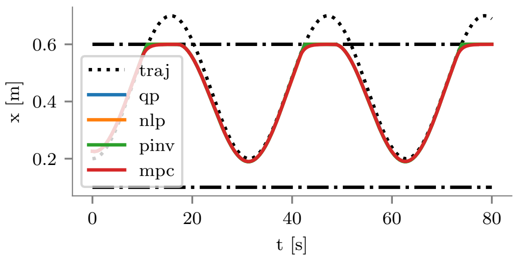


(b) $y$

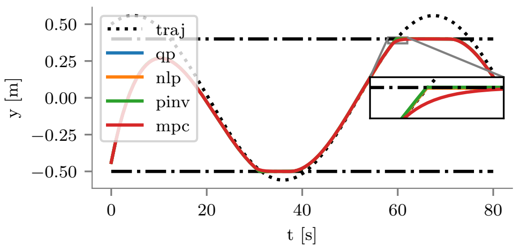


(c) $z$

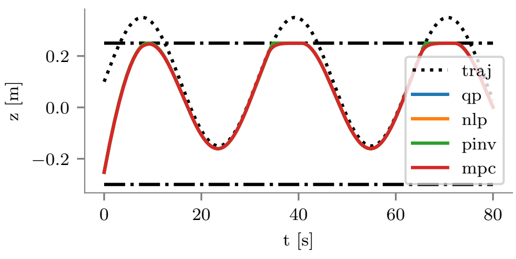
Fig. 5: Position of the end-effector for $x$ and $y$ controlled separately.



(a) $x$



(b) $y$



(c) $z$

Fig. 6: Position of the end-effector for $x$, $y$, and $z$ controlled using a multidimensional constraint. The MPC approach exhibits the exponential approach to the constraint limit.

where $\Delta_t$ is the control duration and $\alpha$ is set to 500. This essentially states that we attempt to maximize the manipulability of the subsequent step. Optimization problems often struggle with square roots, so we square the manipulability index before using it in the cost. As the tracking constraint is of lower priority than the joint limits, we must ensure that the tracking constraint's slack weight is greater than $\alpha$. In Listing 3, we see an example of setting its slack weight to 2000. The QP approach and null-space approach does not support nonlinear costs and do not maximize their manipulability. The MPC has a horizon length of 10 control steps.

In Fig.10 we see the tracking error over time. The lower limit stems from the linearization assumption, and one must either use a path-following approach, or use lower control durations to overcome this. At $t = 30$s, the MPC approach is able to find a different configuration with higher manipulability at the cost of a short duration of deviating from the trajectory.

Listing 3: Equality Constraint With Slack Weight

```
track_cnstr = EqualityConstraint(
    label="tracking_constraint",
    expression=p(q) - p_d(t),
    gain=1.0,
    constraint_type="soft",
    slack_weight=2e3
)
```

This reconfiguration does not affect the final tracking error of the MPC.

In Fig.11 we see the manipulability index $m$ over time. The NLP performs slightly better than the QP approach, and the MPC performs best by far as it chooses to deviate slightly from the trajectory to arrive at a configuration with a higher

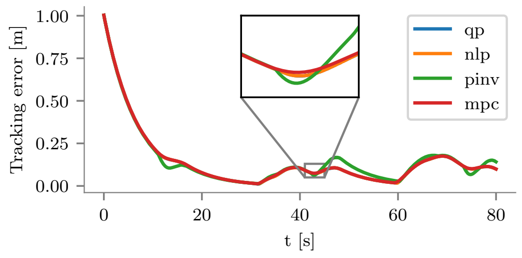


Fig. 7: Tracking error for $x$, $y$, and $z$ as separate constraints. The error does not converge to zero as the desired trajectory goes out of the box.

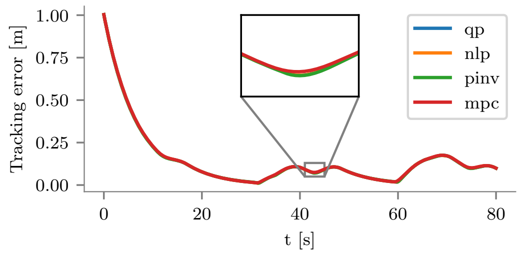


Fig. 8: Tracking error for $x$, $y$ and $z$ as a multidimensional constraint. The error does not converge to zero as the desired trajectory goes outside the box. All are equal except MPC which deviates slightly.

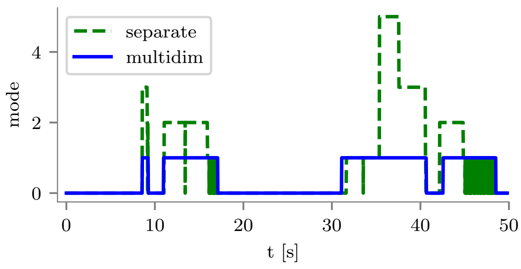


Fig. 9: Excerpt of the mode the null-space based controller is in for both separate constraints and multidimensional constraints. Note the rapid switching at $t = 15$s and $t = 45$s when using separate constraints.

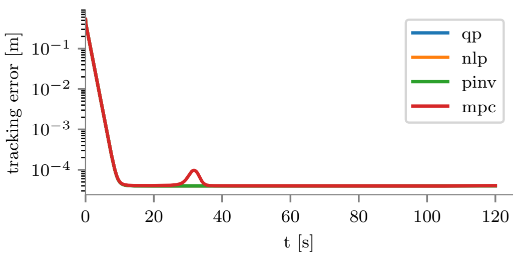


Fig. 10: Tracking error when following the circular trajectory. The MPC approach deviates slightly at $t = 30$s as it is reconfiguring to an orientation with higher manipulability.

manipulability.

In Tab.III we see the initial and average runtimes of the different controllers. The inclusion of the manipulability cost has not made the controllers deviate from the rule of an order of magnitude separation between the approaches.

### D. Input - Tracking a marker in ROS

In this example a UR5 robot tries to track user input. The DefaultRobotInterface is used with ROS and an RViz interactive marker to simulate an external input. The robot is simulated with Gazebo and is controlled at 50 Hz. We use the QP approach in this example. The maximum joint speed is 3 rad/s.

In Fig.12 we see the position of the marker and the end-effector frame. The end-effector has an exponential convergence to the desired input marker position, but as it does not consider the speed of the input marker, there is a tracking error when following the input marker during a continuous motion.

TABLE III: Controller runtimes for Manipulability Example

| | PINV | QP | NLP | MPC |
|---|---|---|---|---|
| Initial | 0.11 ms | 0.48 ms | 4.94 ms | 54.72 ms |
| Average | 0.07 ms | 0.19 ms | 3.02 ms | 49.52 ms |

The DefaultRobotInterface has a delay of 7.8s before it starts as it waits for topics and compiles the controller.

### E. Velocity Equality Constraint - 6 DOF compliance

In this example the end-effector of a UR5 is to comply to forces and torques acting on it. The example uses *urdf2casadi* to determine forward kinematics, and *ur_modern_driver* [30].

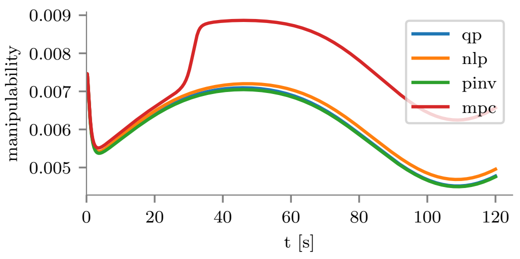


Fig. 11: Manipulability index for different controllers over time when tracking the circular trajectory. The MPC approach is able to find a configuration with higher manipulability index at $t = 30s$.

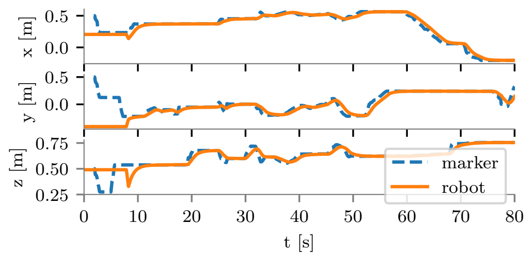


Fig. 12: Position of the input marker and end-effector frame when tracking the input marker.

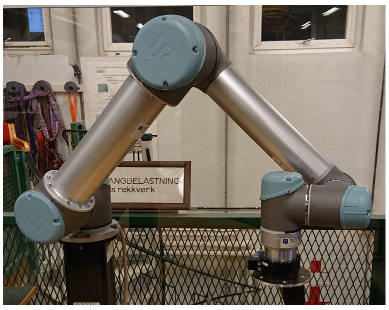


Fig. 13: Experimental setup for 6 DOF compliance.

The end-effector has an ATI Mini45 force/torque sensor attached with a mounting plate on it. The robot is in an open workspace. To ensure the robot does not crash with the table or is moved to undesired regions, the end-effector is limited to a box. The box is defined by the set constraint

$$\begin{bmatrix}-0.7\\-0.4\\-0.2\end{bmatrix} \leq \boldsymbol{p}(\boldsymbol{q}) \leq \begin{bmatrix}-0.3\\0.5\\0.5\end{bmatrix}. \tag{48}$$

Velocity resolved compliance can be achieved using damping control [33] by

$$\begin{bmatrix}\boldsymbol{v}(t)\\\boldsymbol{\omega}(t)\end{bmatrix} = \begin{bmatrix}K_f\boldsymbol{f}(t)\\K_\tau\boldsymbol{\tau}(t)\end{bmatrix} \tag{49}$$

where $\boldsymbol{v}$ is the Cartesian velocity, $\boldsymbol{\omega}$ is the rotational velocity, $\boldsymbol{f}$ are the linear forces, and $\boldsymbol{\tau}$ are the torques. All evaluated at the end-effector. $K_f$ and $K_\tau$ are the damping constants for the forces and torques respectively.

For linear forces $\boldsymbol{f}$ acting on the end-effector, and a position $\boldsymbol{p}(t)$ of the end-effector, we desire:

$$\dot{\boldsymbol{p}}(t) = K_f\boldsymbol{f}^w(t) = K_f\boldsymbol{R}(\boldsymbol{q})\boldsymbol{f}(t) \tag{50}$$

where $\boldsymbol{f}^w$ are the forces acting on the end-effector represented in the world coordinates.

We can relate the rotational velocity to the derivative of the orientation quaternion by following [34], and arrive at

$$\dot{\boldsymbol{Q}}_r(t) = \frac{K_\tau}{2}\bar{\boldsymbol{H}}\left(\begin{bmatrix}\boldsymbol{\tau}(t)\\0\end{bmatrix}\right)\boldsymbol{Q}_r(t) \tag{51}$$

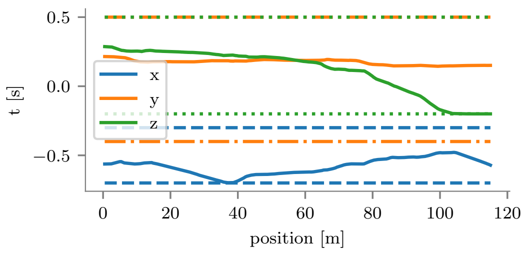


Fig. 14: Position over time of the end-effector during the velocity equality experiment for compliance. The stippled and dotted lines denote the box constraint. From $t = 36$s until $t = 40$s we attempt to pull the end-effector out of the box constraint in $x$ direction, but the set constraint does not allow it. At $t = 80$s we let go of the end-effector, and it drifted to the bottom of the box due to sensor bias.

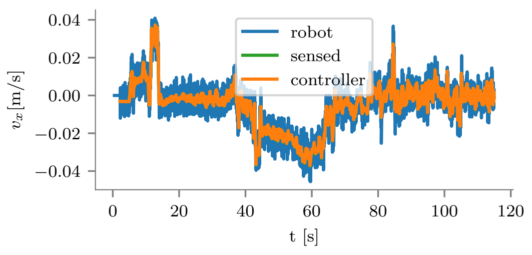
where $\boldsymbol{\tau}(t)$ are the torques acting on the end-effector in the end-effector frame.

The controller is running at 125 Hz, and the force/torque sensor runs at 250 Hz but only the most recent value is used. The damping factors are $K_f = 0.01$ and $K_t = 0.1$. This experiment was run with the QP controller. To inspect the behavior of the system we look at the right hand side of (50) and (51) with the sensor value for force, torque, and $\boldsymbol{q}$. We refer to this as *sensed speed*. The left hand side of (50) and (51) as desired by the CASCLIK controller or as reported by the robot, is referred to as the *controller speed* and the *robot speed* respectively.

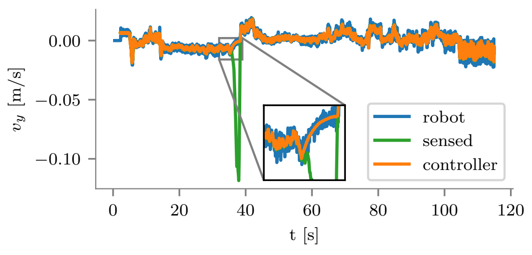
In Fig.14 we see the position of the end-effector over time. In Fig.15 we see sensed speed, controller speed and robot speed. The controller moves to track the target function, resulting in compliance of the end-effector with respect to the force. In Fig.16 we comply with respect to the torques. From $t = 36$s until $t = 40$ we try to pull the end-effector out of the box constraint. As the robot approaches the box constraint, the compliance is reduced to zero. At $t = 80$s we let go of the end-effector and sensor bias moved it slowly to the bottom and along the bottom of the box constraint. The noise is both a result of the latency introduced by using ROS for real-time feedback control, by the computation time of the controller, and by inherent noise in the joint speed signal.

## V. Discussion

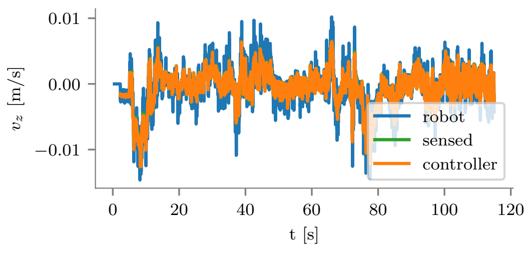
Closed-loop inverse kinematics frameworks handle local problems rather than global problems. When given desired positions far from the current position, closed-loop inverse kinematics may succumb to local minima. This means that they are mid-level controllers to which a desired path may be supplied from a high-level path planner. The model predictive controller formulation is an attempt at bridging the gap between local and global planning. Proper design of cost and constraint formulations to better achieve tasks may lead to better handling of local minima. As of yet, the model predictive approach is



(a) Complying with $f_x$

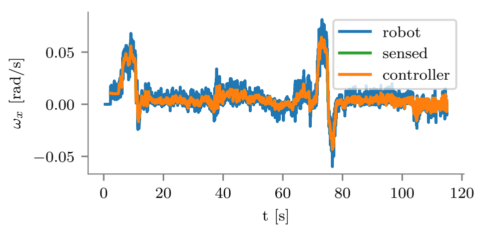


(b) Complying with $f_y$

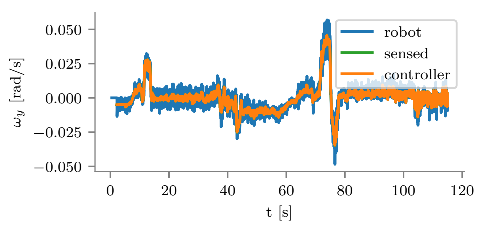


(c) Complying with $f_z$

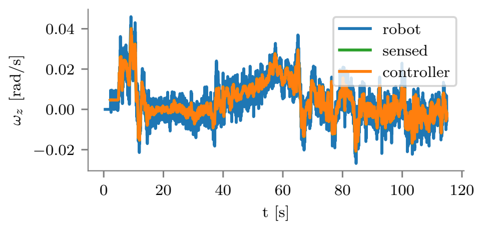
Fig. 15: Sensor and controller Cartesian speeds (right and left hand side of (50)), and robot Cartesian speeds of the end-effector. The zoomed inset at $t = 36$s in subfigure (b) shows the exponential convergence to zero speed in $z$ direction as we try to pull the end-effector out of the box constraint. Otherwise, the sensed and controller speeds perfectly overlap as long as we are inside the box.



(a) Complying with $\tau_x$



(b) Complying with $\tau_y$



(c) Complying with $\tau_z$

Fig. 16: Sensor and controller angular speed (right and left hand side of (51)), and robot angular speeds of the end-effector. As the box constraint only considers the position of the end-effector, not the orientation, the box constraint does not affect the torque compliance.

significantly slower than its reactive counterparts and further work includes investigating sequential quadratic programming approaches with warmstart as they may have shorter execution time than the non-warmstarted interior point method of IPOPT. By using CasADi at the core, CASCLIK can quickly benefit from any new solver implemented in CasADi. The lack of convexity of the constraints along the prediction horizon in the current formulation also suggests that further work should be done to investigate the optimality and stability of the approach.

CASCLIK is independent of the underlying representation used to define kinematics. This allows for inspecting behavior of different representations, but complicates programming for the user. A more robust framework can be created by defining the underlying representation. The choice of using arbitrary functions is a design choice intended to allow a larger range of scenarios and systems to be handled by CASCLIK as well as being easier to implement. Future work includes examining other coordinate representations and constraints.

From the input experiments we see that the controllers have an exponential convergence to the reference signal during positioning, and classical compliance can be implemented in CASCLIK for the controllers that accept velocity equality constraints. As sensors are becoming cheaper and more available, it is important to allow for arbitrary input signals. As the derivative of inputs are unknown, they are considered to be zero. For input signals such as distance sensors or force sensors, this is a false assumption and can lead to tracking error for time varying sensor signals. Future work includes allowing the user to provide input derivatives for CASCLIK. These may come from speed observers, derivative filters, or any other sources the user provides.

CASCLIK only considers control at the velocity setpoint level. This stems from the main use-case for which the framework was intended, industrial manipulators. In most cases, industrial manipulators only provide joint position or joint velocity setpoints. However, the task function approach allows for specifying control at the acceleration or torque level [2]. Extending CASCLIK to include acceleration resolved controllers would allow specifying velocity constraints that ensure convergence to the desired velocity.

From the examples, we see that the null-space approach has similar behavior to the optimization approaches when handling a single set constraint with very high gain. For multiple tasks, the null-space projection operator will cause the set-based task priority framework to behave differently from the optimization approaches. The optimization approaches uses slack variables to handle multiple tasks. This moves prioritization into the cost expression of the optimization problem and does not allow for strict prioritization between tasks, but tuning of the slack weight can be used to specify different behavior of the tasks.

The optimization approaches are closely related, and the complexity of implementing them is similar. The null-space approach requires more bookkeeping by the programmer but generally provides controllers with shorter execution time. Generally the execution speeds are in the order: null-space approach, QP approach, NLP approach, and MPC approach. Each increasing by an order of magnitude in the sequence, depending on the horizon length of the MPC. For rapid prototyping and large set of tasks, the compilation time may also be of interest. As the null-space approach compiles each separate mode, and there are $2^{n_{set}}$ modes, its compilation time drastically increases with multiple set constraints.

The nonlinear cost example shows that the NLP and MPC approach can improve the manipulability in cases where the QP and null-space approach did not. Although the MPC approach managed to find an alternative configuration that increased the manipulability significantly, the reactive NLP approach did not. As the manipulability cost can be added to the QP formulation via Taylor expansion of the cost as in [35], the NLP approach may not be as beneficial as initially expected.

## VI. Conclusion

In this paper, CASCLIK, a rapid prototyping framework for multiple task-based closed-loop inverse kinematics controllers is presented. It translates tasks into quadratic, nonlinear, or model predictive optimization problems that can be solved with CasADi. Tasks are formulated as constraints, and multiple tasks can be achieved simultaneously.

CASCLIK also provides a CasADi based implementation of the set-based task priority inverse kinematics framework. The paper includes a novel multidimensional formulation of the *in tangent cone* function such that the set-based task priority framework can support multidimensional set constraints. The results show that the multidimensional set constraint formulation can give a better representation of the desired behavior than when a multidimensional set constraint is split into multiple one dimensional constraints.

## References

[1] L. Sciavicco and B. Siciliano, "Coordinate Transformation: A Solution Algorithm for One Class of Robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 4, pp. 550–559, jul 1986.

[2] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach*, 1st ed. New York: Oxford University Press, 1991.

[3] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, may 2007.

[4] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and Control of Articulated Robot Arms with Redundancy," *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927–1932, aug 1981.

[5] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[6] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results," *Frontiers in Robotics and AI*, vol. 3, pp. 1–18, apr 2016.

[7] E. Aertbelien and J. De Schutter, "eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, sep 2014, pp. 1540–1546.

[8] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization," Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven, 2013.

[9] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, dec 2014.

[10] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[11] M. H. Arbo, Y. Pane, E. Aertbelien, and W. Deere, "A System Architecture for Constraint-Based Robotic Assembly with CAD Information," in *IEEE 14th International Conference on Automation Science and Engineering (CASE'18)*, aug 2018, pp. 690–696.

[12] N. Mansard, O. Khatib, and A. Kheddar, "A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks," *IEEE Transactions on Robotics (T-RO)*, vol. 25, no. 3, pp. 670–685, 2009.

[13] "Stack-of-tasks." [Online]. Available: http://stack-of-tasks.github.io/

[14] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, jun 2014.

[15] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "iTASC: a tool for multi-sensor integration in robot manipulation," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, aug 2008, pp. 426–433.

[16] "OROCOS: iTaSC wiki." [Online]. Available: http://orocos.org/wiki/orocos/itasc-wiki/2-itasc-software

[17] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *IEEE International Conference on Robotics and Automation (ICRA'03)*, 2003, pp. 2766–2771.

[18] P. Soetens, "RTT: Real-Time Toolkit." [Online]. Available: http://www.orocos.org/rtt

[19] E. Aertbeliën, "expressiongraphs." [Online]. Available: https://github.com/eaertbel/expressiongraph

[20] R. Smits, "KDL: Kinematics and Dynamics Library." [Online]. Available: http://www.orocos.org/kdl

[21] N. Somani, M. Rickert, A. Gaschler, C. Cai, A. Perzylo, and A. Knoll, "Task level robot programming using prioritized non-linear inequality constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16)*, Daejon, oct 2016, pp. 430–437. [Online]. Available: http://ieeexplore.ieee.org/document/7759090/

[22] N. Somani, M. Rickert, and A. Knoll, "An Exact Solver for Geometric Constraints With Inequalities," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 1148–1155, apr 2017.

[23] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, "Intuitive instruction of industrial robots: Semantic process descriptions for small lot production," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'16)*, Daejeon, oct 2016, pp. 2293–2300.

[24] G. Antonelli, G. Indiveri, and S. Chiaverini, “Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems with velocity saturations,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’09)*, oct 2009, pp. 5892–5897.

[25] M. H. Arbo and J. T. Gravdahl, “Stability of the Tracking Problem with Task-Priority Inverse Kinematics,” *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 121–125, 2018.

[26] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.

[27] “A000788: Total number of 1’s in binary expansions of 0, ..., n.” [Online]. Available: https://oeis.org/A000788

[28] A. Colome and C. Torras, “Closed-Loop Inverse Kinematics for Redundant Robots: Comparative Assessment and Two Enhancements,” *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 944–955, apr 2015.

[29] M. H. Arbo, “Mathias Hauan Arbo’s Github page.” [Online]. Available: https://github.com/mahaarbo/

[30] T. T. Andersen, “Optimizing the Universal Robots ROS driver.” Technical University of Denmark, Department of Electrical Engineering, Tech. Rep., 2015.

[31] L. F. Figueredo, B. V. Adorno, J. Y. Ishihara, and G. A. Borges, “Robust kinematic control of manipulator robots using dual quaternion representation,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1949–1955, 2013.

[32] P. M. Larochelle, A. P. Murray, and J. Angeles, “A Distance Metric for Finite Sets of Rigid-Body Displacements via the Polar Decomposition,” *Journal of Mechanical Design*, vol. 129, no. 8, p. 883, 2007.

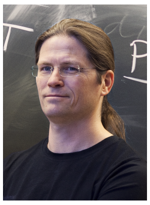
[33] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008.

[34] X. Wang, D. Han, C. Yu, and Z. Zheng, “The geometric structure of unit dual quaternion with application in kinematic control,” *Journal of Mathematical Analysis and Applications*, vol. 389, no. 2, pp. 1352–1364, 2012.

[35] K. Dufour and W. Suleiman, “On integrating manipulability index into inverse kinematics solver,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’17)*, Vancouver, sep 2017, pp. 6967–6972.

**Mathias Hauan Arbo** received the M. S. degree in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway in 2015, where he is currently pursuing the Ph.D. degree. He received the best student paper award from the 14th IEEE Conference on Automation Science and Engineering (CASE) in 2018. His research interests include robotic assembly, optimization-based control, force control, and motion control.

**Esten Ingar Grötli** received the M. S. degree in Engineering Cybernetics and the Ph.D. degree in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2005 and 2010 respectively. From 2006 until 2007 he was a visiting Ph.D. student at the University of California, Berkely, California, USA. He was a Postdoc researcher at the Norwegian University of Science and Technology (NTNU) from 2010 until 2014, and a lecturer at the Royal Norwegian Air Force Academy, Kuhaugen, Norway, from 2012 until 2013. He is currently a senior researcher at SINTEF DIGITAL, Trondheim, Norway. His research interests include estimation, sensor fusion, motion planning, data analysis, and autonomy.



**Jan Tommy Gravdahl** received the Siv.ing and Dr.ing degrees in Engineering Cybernetics from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 1994 and 1998, respectively. He was appointed Associate Professor (2001) and Professor (2005) at the Department of Engineering Cybernetics, NTNU, where he also served as Head of Department in 2008-09. In 2007–08, he was a Visiting Professor at The Centre for Complex Dynamic Systems and Control (CDSC), The University of Newcastle, Australia. He has supervised the graduation of more than 100 MSc in addition to 13 PhD candidates. He has published five books and more than 200 papers in international conferences and journals. In 2000 and 2017, he was awarded the IEEE Transactions on Control Systems Technology Outstanding Paper Award. He is a senior editor of the IFAC journal Mechatronics. His current research interests include mathematical modeling and nonlinear control in general, in particular applied to turbomachinery, marine vehicles, spacecraft, robots, and high-precision mechatronic systems.

# References

[1] M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl, "On the globally exponentially convergent immersion and invariance speed observer for mechanical systems", in *2017 American Control Conference (ACC)*, Seattle, May 2017, pp. 3294–3299. DOI: `10.23919/ACC.2017.7963455`.

[2] ——, "On model predictive path following and trajectory tracking for industrial robots", in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Xi'an, Aug. 2017, pp. 100–105. DOI: `10.1109/COASE.2017.8256087`.

[3] ——, "Mid-Level MPC and 6 DOF output path following for robotic manipulators", in *IEEE Conference on Control Technology and Applications (CCTA'17)*, Mauna Lani, Aug. 2017, pp. 450–456. DOI: `10.1109/CCTA.2017.8062503`.

[4] M. H. Arbo, Y. Pane, E. Aertbeliën, and W. Decré, "A System Architecture for Constraint-Based Robotic Assembly with CAD Information", in *IEEE 14th International Conference on Automation Science and Engineering (CASE'18)*, Munich, Aug. 2018, pp. 690–696. DOI: `10.1109/COASE.2018.8560450`.

[5] M. H. Arbo and J. T. Gravdahl, "Stability of the Tracking Problem with Task-Priority Inverse Kinematics", *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 121–125, 2018. DOI: `10.1016/j.ifacol.2018.11.528`.

[6] M. H. Arbo, *CASCLIK webpage*. [Online]. Available: `https://github.com/mahaarbo/casclik` (visited on 11/28/2018).

[7] ——, *Urdf2casadi*. [Online]. Available: `https://github.com/mahaarbo/urdf2casadi` (visited on 01/10/2018).

[8] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization", Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven, 2013.

[9] M. H. Arbo, *Arbench*. [Online]. Available: `https://github.com/mahaarbo/arbench` (visited on 01/10/2018).

[10] J. Riegel and Y. van Havre, *FreeCAD: Parametric 3D modeler*. [Online]. Available: `https://www.freecadweb.org` (visited on 02/06/2018).

[11] I. Eriksen and M. H. Arbo, *KUKA KVP Hardware Interface*. [Online]. Available: `https://github.com/itk-thrivaldi/kuka_kvp_hw_interface` (visited on 01/10/2018).

[12] M. H. Arbo, T. Utstumo, E. Brekke, and J. T. Gravdahl, "Unscented Multi-Point Smoother for Fusion of Delayed Displacement Measurements: Application to Agricultural Robots", *Modeling, Identification and Control: A Norwegian Research Bulletin (MIC)*, vol. 38, no. 1, pp. 1–9, 2017. DOI: `10.4173/mic.2017.1.1`.

[13] J. Westbrook, "30% Iron Chef", *Futurama*, vol. 3, no. 22, 2002.

[14] S. Y. Nof, Ed., *Handbook of Industrial Robotics*, 2nd. Hoboken, NJ: John Wiley & Sons, Inc., Feb. 1999. DOI: `10.1002/9780470172506`.

[15] J. F. Engelberger, *Robotics in Practice*. Boston, MA: Springer US, 1983. DOI: `10.1007/978-1-4684-7120-5`.

[16] G. C. J. Devol, "US2988237A: Programmed Article Transfer", *United States Patent Office*, 1961.

[17] IRobot, *iRobot Roomba*. [Online]. Available: `https://www.irobot.com/for-the-home/vacuuming/roomba` (visited on 01/10/2018).

[18] *Eelume*. [Online]. Available: `https://eelume.com/` (visited on 01/11/2018).

[19] NASA, *Mars Science Laboratory: Curiosity Rover*. [Online]. Available: `https://mars.jpl.nasa.gov/msl/` (visited on 01/10/2018).

[20] Intuitive Surgical, *Da Vinci Surgical System*. [Online]. Available: `https://www.intuitive.com/en/products-and-services/da-vinci` (visited on 01/10/2018).

[21] National Institute of Standards and Technology, *Wikimedia: Equipment in the Automated Manufacturing Research Facility*, 2014. [Online]. Available: `https://commons.wikimedia.org/wiki/File:AutomatedManufacturingResearchFacility_011.jpg` (visited on 12/22/2018).

[22] L. G. Jøssang, *Frå Jærmuseet sine samlingar: Trallfa Robot TR 2000*, Nærbø, 2005. [Online]. Available: `https://www.jaermuseet.no/samlingar/wp-content/uploads/sites/16/2011/06/2005.7-Trallfa-robot-TR2000.pdf` (visited on 12/20/2018).

[23] L. Westerlund, *The Extended Arm of Man - A History of the Industrial Robot*. Informationsförlaget, 2000, ISBN: 978-9177364672.

[24] P. C. Watson, "US05732286: Remote center compliance system", *United States Patent Office*, 1976.

[25] B. Shimano, C. Geschke, and C. Spalding, “VAL-II: A new robot control system for automatic manufacturing”, in *IEEE International Conference on Robotics and Automation (ICRA’84)*, vol. 1, Atlanta, 1984, pp. 278–292. DOI: `10.1109/ROBOT.1984.1087187`.

[26] H. Bruyninckx, “Open robot control software: the OROCOS project”, in *IEEE International Conference on Robotics and Automation (ICRA’01)*, vol. 3, Seoul, 2001, pp. 2523–2528. DOI: `10.1109/ROBOT.2001.933002`.

[27] *The OROCOS Project*. [Online]. Available: `http://www.orocos.org/` (visited on 12/23/2018).

[28] M. Morgan Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System”, in *IEEE International Conference on Robotics and Automation (ICRA’09) Workshop on Open Source Robotics*, Kobe, May 2009.

[29] *ROS: Robot Operating System*. [Online]. Available: `http://www.ros.org` (visited on 12/23/2018).

[30] B. Gerkey, *IEEE Spectrum: ROS, the Robot Operating System, Is Growing Faster Than Ever, Celebrates 8 Years*, 2015. [Online]. Available: `https://spectrum.ieee.org/automaton/robotics/robotics-software/ros-robot-operating-system-celebrates-8-years` (visited on 12/23/2018).

[31] German Aerospace Center (DLR), *History of the DLR LWR*. [Online]. Available: `https://www.dlr.de/rm/en/desktopdefault.aspx/tabid-12464/21732_read-44586/` (visited on 12/23/2018).

[32] Universal Robots, *History of Universal Robots*. [Online]. Available: `https://www.universal-robots.com/about-universal-robots/our-history/` (visited on 12/23/2018).

[33] ABB, *ABB’s Yumi Collaborative robot named "2016 Best Industrial Robot"*, 2016. [Online]. Available: `http://www.abb.com/cawp/seitp202/C9AA2AC92A152904C125801200537DF0.aspx` (visited on 01/15/2019).

[34] E. Guizzo and E. Ackerman, *IEEE Spectrum: How Rethink Robotics Built Its New Baxter Robot Worker*, 2012. [Online]. Available: `https://spectrum.ieee.org/robotics/industrial-robots/rethink-robotics-baxter-robot-factory-worker` (visited on 12/23/2018).

[35] ABB, *ABB introduces YuMi®, world’s first truly collaborative dual-arm robot*, 2015. [Online]. Available: `http://www.abb.com/cawp/seitp202/33a7cccb0886548048257e2600295564.aspx` (visited on 12/23/2018).

[36] J. Vanian, *Fortune: The Multi-Billion Dollar Robotics Market Is About to Boom*, 2016. [Online]. Available: `http://fortune.com/2016/02/24/robotics-market-multi-billion-boom/` (visited on 12/20/2018).

[37] H. L. Sirkin, M. Zinser, and J. Rose, “Boston Consulting Group: The Robotics Revolution”, *BCG perspectives*, vol. 9, pp. 1–25, 2015.

[38] L. E. Kavraki and M. N. Kolountzakis, “Partitioning a planar assembly into two connected parts is NP-complete”, *Information Processing Letters*, vol. 55, no. 3, pp. 159–165, Aug. 1995. DOI: 10.1016/0020-0190(95)00083-O.

[39] S. Elliott Fahlman, “A planning system for robot construction tasks”, *Artificial Intelligence*, vol. 5, no. 1, pp. 1–49, 1974. DOI: 10.1016/0004-3702(74)90008-3.

[40] L. I. Lieberman and M. A. Wesley, “AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly”, *IBM Journal of Research and Development*, vol. 21, no. 4, pp. 321–333, Jul. 1977. DOI: 10.1147/rd.214.0321.

[41] T. Lozano-Pérez and P. H. Winston, “LAMA: a language for automatic mechanical assembly”, in *5th International Joint Conference on Artificial Intelligence (IJCAI’77)*, Cambridge, 1977, pp. 710–716.

[42] R. J. Popplestone, A. P. Ambler, and I. Bellos, “RAPT: a Language for Describing Assemblies.”, *Industrial Robot*, vol. 5, no. 3, pp. 131–137, 1978. DOI: 10.1108/eb004501.

[43] T. Lozano-Pérez and R. A. Brooks, “An approach to automatic robot programming”, in *ACM 14th annual conference on Computer science - (CSC ’86)*, Cincinnati: ACM Press, 1986, pp. 61–69. DOI: 10.1145/324634.325195.

[44] S. G. Kaufman, R. H. Wilson, R. E. Jones, T. L. Calton, and A. L. Ames, “The Archimedes 2 mechanical assembly planning system”, in *IEEE International Conference on Robotics and Automation (ICRA’96)*, vol. 4, Minneapolis, 1996, pp. 3361–3368. DOI: 10.1109/ROBOT.1996.509225.

[45] Adept Technology Inc., *V + Language User ’s Guide v.12.1*. Adept Technoly Inc., 1997.

[46] U. Thomas and F. Wahl, “A system for automatic planning, evaluation and execution of assembly sequences for industrial robots”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’01)*, vol. 3, Maui: IEEE, 2001, pp. 1458–1464. DOI: 10.1109/IROS.2001.977186.

[47] H. Mosemann and F. Wahl, “Automatic decomposition of planned assembly sequences into skill primitives”, *IEEE Transactions on Robotics and Automation*, vol. 17, no. 5, pp. 709–718, 2001. DOI: 10.1109/70.964670.

[48] U. Thomas, M. Barrenscheen, and F. Wahl, “Efficient assembly sequence planning using stereographical projections of C-space obstacles”, in *IEEE International Symposium on Assembly and Task Planning (ISATP’03)*, Besancon, 2003, pp. 96–102. DOI: 10.1109/ISATP.2003.1217194.

[49] U. Thomas, F. Wahl, J. Maass, and J. Hesselbach, “Towards a new concept of robot programming in high speed assembly applications”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’05)*, Edmonton, 2005, pp. 3827–3833. DOI: 10.1109/IROS.2005.1545582.

[50] U. Thomas, S. Molkenstruck, R. Iser, and F. M. Wahl, “Multi Sensor Fusion in Robot Assembly Using Particle Filters”, in *IEEE International Conference on Robotics and Automation (ICRA’07)*, Roma, Apr. 2007, pp. 3837–3843. DOI: 10.1109/ROBOT.2007.364067.

[51] U. Thomas and F. M. Wahl, “Assembly Planning and Task Planning — Two Prerequisites for Automated Robot Programming”, in *Springer Tracts in Advanced Robotics*, vol. 67, 2010, pp. 333–354. DOI: 10.1007/978-3-642-16785-0_19.

[52] T. Lozano-Perez, “Robot programming”, *Proceedings of the IEEE*, vol. 71, no. 7, pp. 821–841, 1983. DOI: 10.1109/PROC.1983.12681.

[53] A. Ambler and R. Popplestone, “Inferring the positions of bodies from specified spatial relationships”, *Artificial Intelligence*, vol. 6, no. 2, pp. 157–174, Jun. 1975. DOI: 10.1016/0004-3702(75)90007-7.

[54] M. Kim and C.-H. Wu, “A formal part mating model for generating compliance control strategies of assembly operations”, in *IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings (SMC’90)*, Los Angeles, 1990, pp. 611–616. DOI: 10.1109/ICSMC.1990.142185.

[55] A. Bourjault, “Contribution a une approache methodologique de l’assemblage automatise: elaboration automatique des sequence s operatories”, Ph.D. Thesis, L’Université de Franche-Comté, 1984.

[56] T. De Fazio and D. Whitney, “Simplified generation of all mechanical assembly sequences”, *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 640–658, Dec. 1987. DOI: 10.1109/JRA.1987.1087132.

[57] L. Homem de Mello and A. Sanderson, “AND/OR graph representation of assembly plans”, *IEEE Transactions on Robotics and Automation*, vol. 6, no. 2, pp. 188–199, Apr. 1990. DOI: 10.1109/70.54734.

[58] D. E. Whitney, *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*. Oxford University Press, 2004, ISBN: 978-0195157826.

[59] K. Ikeuchi and S. B. Kang, “Assembly Plan from Observation”, *Technical Report FS-93-04, AAAI press*, 1993.

[60] SARAFun, *Smart Assembly with Advanced Functionalities*. [Online]. Available: http://h2020sarafun.eu/ (visited on 01/10/2018).

[61] N. Wiener, *Cybernetics: Or the Control and Communication in the Animal and the Machine*, 2nd Editio. Martino Fine Books.

[62] K. B. Hammersvik, “Her er de tryggeste utdanningsvalgene”, *NRK Livsstil*, Apr. 2018. [Online]. Available: https://www.nrk.no/livsstil/her-er-de-tryggeste-utdanningsvalgene-1.14003297.

[63] S. Fölster, “Vartannat jobb automatiseras inom 20 år”, Stiftelsen för strategisk forskning, Tech. Rep., 2014. [Online]. Available: http://strategiska.se/app/uploads/varannat-jobb-automatiseras.pdf.

[64] C. B. Frey and M. A. Osborne, “The future of employment: How Susceptible are jobs to computerisation?”, Oxford Martin School, Oxford, England, Tech. Rep., 2013.

[65] ——, “The future of employment: How susceptible are jobs to computerisation?”, *Technological Forecasting and Social Change*, vol. 114, pp. 254–280, Jan. 2017. DOI: `10.1016/j.techfore.2016.08.019`.

[66] National Center for O*NET Development, *O*NET Resource Center*. [Online]. Available: `https://www.onetcenter.org/` (visited on 12/23/2018).

[67] J. Manyika, M. Chui, M. Miremadi, J. Bughin, K. George, P. Willmott, and M. Dewhurst, “A future that works: Automation, employment, and productivity”, *Mckinsey Global Institute*, 2017.

[68] Mckinsey Global Institute, *Where machines could replace humans - and where they can’t (yet)*, 2017. [Online]. Available: `https://public.tableau.com/profile/mckinsey.analytics%7B%5C#%7D!/vizhome/InternationalAutomation/WhereMachinesCanReplaceHumans` (visited on 12/23/2018).

[69] E. Brynjolfsson and A. Mcafee, *The Second Machine Age*. New York, NY: W. W. Norton & Company, Inc., 2014, ISBN: 978-0-393-35064-7.

[70] M. Andreessen, *Wall Street Journal - Why Software Is Eating The World*, 2011. [Online]. Available: `https://www.wsj.com/articles/SB10001424053111903480904576512250915629460` (visited on 01/10/2019).

[71] *Digitale Wirtschaft und Gesellschaft: Industrie 4.0*. [Online]. Available: `https://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html` (visited on 12/23/2018).

[72] M. A. Kamarul Bahrin, M. F. Othman, N. H. Nor Azli, and M. F. Talib, “INDUSTRY 4.0: A REVIEW ON INDUSTRIAL AUTOMATION AND ROBOTIC”, *Jurnal Teknologi*, vol. 78, no. 6-13, 2016. DOI: `10.11113/jt.v78.9285`.

[73] DigitalNorway, *Digital21*. [Online]. Available: `https://digital21.no/` (visited on 12/23/2018).

[74] H. Sinding-Larsen, “Ingen mennesker er tjent med slavearbeid”, *Aftenposten*, pp. 5–6, 8. January, 1966.

[75] The Apache Software Foundation, *About the Apache HTTP Server Project*. (visited on 12/23/2018).

[76] B. McCullough, *On The 20th Anniversary - An Oral History of Netscape’s Founding*. [Online]. Available: `http://www.internethistorypodcast.com/2014/04/on-the-20th-anniversary-an-oral-history-of-netscapes-founding/` (visited on 01/10/2018).

[77] Franka Emika, *Franka Emika: libfranka documentation*. [Online]. Available: `https://frankaemika.github.io/docs/libfranka.html` (visited on 01/08/2019).

[78] G. Schreiber, A. Stemmer, and R. Bischoff, “The fast research interface for the kuka lightweight robot”, in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA’10)*, 2010, pp. 15–21.

[79] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, p. 1611. DOI: `10.1007/978-3-540-30301-5`.

[80] A. C. Bittencourt, “Modeling and Diagnosis of Friction and Wear in Industrial Robots”, PhD-Thesis, Linköping University, 2014, pp. 1–232, ISBN: 9781467366953.

[81] H. Asada, T. Kanade, and I. Takeyama, “Control of a Direct-Drive Arm”, *Journal of Dynamic Systems, Measurement, and Control*, vol. 105, no. 3, p. 136, 1983. DOI: `10.1115/1.3140645`.

[82] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, “The DLR lightweight robot: design and control concepts for robots in human environments”, *Industrial Robot: An International Journal*, vol. 34, no. 5, C. Loughlin, Ed., pp. 376–385, Aug. 2007. DOI: `10.1108/01439910710774386`.

[83] S. Ozgoli and H. D. Taghirad, “A Survey on the Control of Flexible Joint Robots”, *Asian Journal of Control*, vol. 8, no. 4, pp. 332–344, 2008. DOI: `10.1111/j.1934-6093.2006.tb00285.x`.

[84] B. Siciliano, “Control in robotics: open problems and future directions”, in *IEEE International Conference on Control Applications (CCA’98)*, vol. 1, Trieste, 1998, pp. 81–85. DOI: `10.1109/CCA.1998.728949`.

[85] A. Y. C. Nee, Ed., *Handbook of Manufacturing Engineering and Technology*. London: Springer London, 2013. DOI: `10.1007/978-1-4471-4976-7`.

[86] E. Hedberg, M. Norrlöf, S. Moberg, and S. Gunnarsson, “Comparing Feedback Linearization and Jacobian Linearization for LQ Control of an Industrial Manipulator”, in *12th IFAC Symposium on Robot Control (SYROCO’18)*, Budapest, 2018.

[87] A. Stolt, “On Robotic Assembly using Contact Force Control and Estimation”, Ph.D. Thesis, Lund University, 2015, ISBN: 9789176234563.

[88] M. Lind, J. Schrimpf, and T. Ulleberg, “Open Real-Time Robot Controller Framework”, in *3rd CIRP Conference on Assembly Technology and Systems (CATS’10)*, Trondheim, 2010, pp. 13–18.

[89] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation”, *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987. DOI: `10.1109/JRA.1987.1087068`.

[90] N. Hogan, “Impedance control of industrial robots”, *Robotics and Computer-Integrated Manufacturing*, vol. 1, no. 1, pp. 97–113, Jan. 1984. DOI: 10.1016/0736-5845(84)90084-X.

[91] M. T. Mason, “Compliance and Force Control for Computer Controlled Manipulators”, *IEEE Transactions on Systems, Man, and Cybernetics (T-SMC)*, vol. 11, no. 6, pp. 418–432, 1981. DOI: 10.1109/TSMC.1981.4308708.

[92] S. Eppinger and W. Seering, “Introduction to dynamic models for robot force control”, *IEEE Control Systems Magazine*, vol. 7, no. 2, pp. 48–52, 1987. DOI: 10.1109/MCS.1987.1105274.

[93] E. Aertbeliën and J. De Schutter, “eTaSL/eTC: A constraint-based task specification language and robot controller using expression graphs”, in *IEEE International Conference on Intelligent Robots and Systems (IROS’14)*, Chicago, Sep. 2014, pp. 1540–1546. DOI: 10.1109/IROS.2014.6942760.

[94] T. Kröger, *On-Line Trajectory Generation in Robotic Systems*, ser. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer, 2010, vol. 58. DOI: 10.1007/978-3-642-05175-3.

[95] M. Norrlöf and S. Gunnarsson, “Time and frequency domain convergence properties in iterative learning control”, *International Journal of Control*, vol. 75, no. 14, pp. 1114–1126, Jan. 2002. DOI: 10.1080/00207170210159122.

[96] T. Faulwasser, B. Kern, and R. Findeisen, “Model predictive path-following for constrained nonlinear systems”, in *48th IEEE Conference on Decision and Control (CDC’09) held jointly with 28th Chinese Control Conference (CCC’09)*, Shanghai, Dec. 2009, pp. 8642–8647. DOI: 10.1109/CDC.2009.5399744.

[97] T. Faulwasser, “Optimization-based solutions to constrained trajectory-tracking and path-following problems”, Ph.D. Thesis, Otto Von Guericke University Magdeburg, Magdeburg, 2013. DOI: 10.2370/9783844015942.

[98] D. E. Whitney, “Resolved Motion Rate Control of Manipulators and Human Prostheses”, *IEEE Transactions on Man Machine Systems*, vol. 10, no. 2, pp. 47–53, Jun. 1969. DOI: 10.1109/TMMS.1969.299896.

[99] N. Mansard, O. Khatib, and A. Kheddar, “A Unified Approach to Integrate Unilateral Constraints in the Stack of Tasks”, *IEEE Transactions on Robotics (T-RO)*, vol. 25, no. 3, pp. 670–685, 2009. DOI: 10.1109/TRO.2009.2020345.

[100] C. A. Klein and C.-H. Huang, “Review of pseudoinverse control for use with kinematically redundant manipulators”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 2, pp. 245–250, 1983. DOI: 10.1109/TSMC.1983.6313123.

[101] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators", *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997. DOI: 10.1109/70.585902.

[102] J. M. Hollerbach and K. C. Suh, "Redundancy resolution of manipulators through torque optimization", *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 308–316, 1987. DOI: 10.1109/JRA.1987.1087111.

[103] A. Balestrino, G. De Maria, and L. Sciavicco, "Robust Control of Robotic Manipulators", *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 2435–2440, Jul. 1984. DOI: 10.1016/S1474-6670(17)61347-8.

[104] L. Sciavicco and B. Siciliano, "Coordinate Transformation: A Solution Algorithm for One Class of Robots", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 4, pp. 550–559, Jul. 1986. DOI: 10.1109/TSMC.1986.289258.

[105] H. Das, J.-J. E. Slotine, and T. B. Sheridan, "Inverse kinematic algorithms for redundant systems", in *IEEE International Conference on Robotics and Automation (ICRA'88)*, Philadelphia, 1988, pp. 43–48. DOI: 10.1109/ROBOT.1988.12021.

[106] C. Samson, M. Le Borgne, and B. Espiau, *Robot Control: The Task Function Approach*, 1st Editio. New York: Oxford University Press, 1991.

[107] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results", *Frontiers in Robotics and AI*, vol. 3, pp. 1–18, Apr. 2016. DOI: 10.3389/frobt.2016.00016.

[108] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analysis and Control of Articulated Robot Arms with Redundancy", *IFAC Proceedings Volumes*, vol. 14, no. 2, pp. 1927–1932, Aug. 1981. DOI: 10.1016/S1474-6670(17)63754-6.

[109] G. Antonelli, G. Indiveri, and S. Chiaverini, "Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems with velocity saturations", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, St. Louis, Oct. 2009, pp. 5892–5897. DOI: 10.1109/IROS.2009.5354636.

[110] M. C. Bjerkeng, "Sensor-Based Control of Industrial Manipulators", Ph.D. Thesis, Norwegian University of Science and Technology (NTNU), 2013.

[111] M. C. Bjerkeng, P. Falco, C. Natale, and K. Y. Pettersen, "Discrete-time stability analysis of a control architecture for heterogeneous robotic systems", in *IEEE International Conference on Intelligent Robots and Systems (IROS'13)*, Tokyo, 2013, pp. 4778–4783. DOI: 10.1109/IROS.2013.6697045.

[112] ——, “Stability Analysis of a Hierarchical Architecture for Discrete-Time Sensor-Based Control of Robotic Systems”, *IEEE Transactions on Robotics (T-RO)*, vol. 30, no. 3, pp. 745–753, Jun. 2014. DOI: 10.1109/TRO.2013.2294882.

[113] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots”, in *IEEE International Conference on Robotics and Automation (ICRA’85)*, vol. 2, St. Louis, 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[114] B. Faverjon and P. Tournassoud, “A local based approach for path planning of manipulators with a high number of degrees of freedom”, in *IEEE International Conference on Robotics and Automation (ICRA’87)*, vol. 4, Raleigh, 1987, pp. 1152–1159. DOI: 10.1109/ROBOT.1987.1087982.

[115] H. Kazerooni, T. Sheridan, and P. Houpt, “Robust compliant motion for manipulators, part I: The fundamental concepts of compliant motion”, *IEEE Journal on Robotics and Automation*, vol. 2, no. 2, pp. 83–92, 1986. DOI: 10.1109/JRA.1986.1087045.

[116] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, “Constraint-based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty”, *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, May 2007. DOI: 10.1177/027836490707809107.

[117] M. Klotzbücher and H. Bruyninckx, “Coordinating Robotic Tasks and Systems with rFSM Statecharts”, *Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 28–56, 2012.

[118] D. Schütz and F. M. Wahl, Eds., *Robotic Systems for Handling and Assembly*, ser. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer, 2011, vol. 67. DOI: 10.1007/978-3-642-16785-0.

[119] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, “A new skill based robot programming language using UML/P Statecharts”, in *IEEE International Conference on Robotics and Automation (ICRA’13)*, Karlsruhe, May 2013, pp. 461–466. DOI: 10.1109/ICRA.2013.6630615.

[120] F. Proctor, S. Balakirsky, Z. Kootbally, T. Kramer, C. Schlenoff, and W. Shackleford, “The Canonical Robot Command Language (CRCL)”, *Industrial Robot*, vol. 43, no. 5, pp. 495–502, 2016. DOI: 10.1108/IR-01-2016-0037.

[121] T. Faulwasser and R. Findeisen, “Nonlinear Model Predictive Control for Constrained Output Path Following”, *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 1026–1039, Apr. 2016. DOI: 10.1109/TAC.2015.2466911.

[122] S. M. LaValle, *Planning algorithms*. New York, NY: Cambridge University Press, 2006.

[123] Z. Shiller and S. Dubowsky, “On computing the global time-optimal motions of robotic manipulators in the presence of obstacles”, *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 785–797, 1991. DOI: 10.1109/70.105387.

[124] J. Vannoy and Jing Xiao, “Real-Time Adaptive Motion Planning (RAMP) of Mobile Manipulators in Dynamic Environments With Unforeseen Changes”, *IEEE Transactions on Robotics (T-RO)*, vol. 24, no. 5, pp. 1199–1212, Oct. 2008. DOI: 10.1109/TRO.2008.2003277.

[125] R. Katzschmann, T. Kroger, T. Asfour, and O. Khatib, “Towards online trajectory generation considering robot dynamics and torque limits”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’13)*, Tokyo, Nov. 2013, pp. 5644–5651. DOI: 10.1109/IROS.2013.6697174.

[126] O. Brock and L. Kavraki, “Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces”, in *IEEE International Conference on Robotics and Automation (ICRA’01)*, vol. 2, Seoul, 2001, pp. 1469–1474. DOI: 10.1109/ROBOT.2001.932817.

[127] O. Brock and O. Khatib, “Elastic Strips: A Framework for Motion Generation in Human Environments”, *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, Dec. 2002. DOI: 10.1177/0278364902021012002.

[128] T. Mercy, E. Hostens, and G. Pipeleers, “Online motion planning for autonomous vehicles in vast environments”, in *IEEE 15th International Workshop on Advanced Motion Control (AMC’18)*, Tokyo, Mar. 2018, pp. 114–119. DOI: 10.1109/AMC.2019.8371072.

[129] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. DOI: 10.1109/70.508439.

[130] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning”, in *IEEE International Conference on Robotics and Automation (ICRA’09)*, Kobe, May 2009, pp. 489–494. DOI: 10.1109/ROBOT.2009.5152817.

[131] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning”, in *IEEE International Conference on Robotics and Automation (ICRA’11)*, Shanghai, May 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[132] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, “Real-time trajectory generation using model predictive control”, in *IEEE International Conference on Automation Science and Engineering (CASE’15)*, vol. 147, Gothenburg, Aug. 2015, pp. 942–948. DOI: 10.1109/CoASE.2015.7294220.

[133] S. Chitta and I. A. Sucan, *MoveIt!* [Online]. Available: `http://moveit.ros.org/` (visited on 01/09/2019).

[134] A. Loria, “Observers are Unnecessary for Output-Feedback Control of Lagrangian Systems”, *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 905–920, Apr. 2016. DOI: `10.1109/TAC.2015.2446831`.

[135] N. Somani, M. Rickert, A. Gaschler, C. Cai, A. Perzylo, and A. Knoll, “Task level robot programming using prioritized non-linear inequality constraints”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’16)*, Daejon, Oct. 2016, pp. 430–437. DOI: `10.1109/IROS.2016.7759090`.

[136] A. Perzylo, N. Somani, S. Profanter, I. Kessler, M. Rickert, and A. Knoll, “Intuitive instruction of industrial robots: Semantic process descriptions for small lot production”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’16)*, Daejeon, Oct. 2016, pp. 2293–2300. DOI: `10.1109/IROS.2016.7759358`.

[137] K. Dufour and W. Suleiman, “On integrating manipulability index into inverse kinematics solver”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’17)*, Vancouver, Sep. 2017, pp. 6967–6972. DOI: `10.1109/IROS.2017.8206621`.

[138] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen, and J. T. Gravdahl, “Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks”, in *IEEE Conference on Control Technology and Applications (CCTA’17)*, Mauna Lani, Aug. 2017, pp. 142–149. DOI: `10.1109/CCTA.2017.8062454`.

[139] G. Antonelli, F. Arrichiello, and S. Chiaverini, “The Entrapment/Escorting Mission”, *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 22–29, Mar. 2008. DOI: `10.1109/M-RA.2007.914932`.

[140] Acutronic Robotics, *Hardware Robot Information Model*. [Online]. Available: `https://acutronicrobotics.com/technology/hrim/` (visited on 01/10/2018).

[141] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator”, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’04)*, vol. 3, Sendai, 2004, pp. 2149–2154. DOI: `10.1109/IROS.2004.1389727`.