Constraint-based Dual Arm Control for Automated Wiring of Electrical Cabinets

Lorenz Halt* and Philipp Tenbrock*

***Abstract*— In this application we explore a new approach to automated wire handling in order to decrease longterm maintenance efforts and speed up development — reducing specialized hardware to a minimum, while handling the complexity with software. A high reuse rate of previously developed and thoroughly tested robotic skills minimized the risk of software bugs. The constraint-based programming approach made manual transformations between the task space of the previously planned wiring map and the robot operation space obsolete and reduced the overall development time dramatically.**

I. Introduction and Project

Handling flexible and pliable materials is a challenging task in automated production. Often, complex and specialized tooling is used to allow predictions about the materials by mechanically restricting or guiding during handling. In most cases, a human worker is required if the uncertainties cannot be limited mechanically.

Cables and single wires fall into this category of difficult materials for automated manipulation. Electrical cabinets are good examples of highly customizable industrial products, for which automation suffers under the complexity of the handled materials. A typical electrical cabinet consists of a base plate with attached electrical components and interconnecting cable ducts. Based on the actual configuration individual components are connected with wires running within the cable ducts, respectively. Both, the electrical component configuration and a map of all wires can be virtually planned with CAD tools and automatically optimized. However, the actual process of wiring is a manual operation.

In this application we explored a new approach to automated wire handling — reducing specialized hardware to a minimum while handling the complexity of the application with software.

By following this paradigm, we want to reduce the longterm hardware maintenance efforts and speed up development by a) prototyping and testing with standardized hardware that is easily duplicatable and b) fast reconfigurability because software is updated instead of hardware.

However, the psychological drawbacks of the approach are a) the potential underestimation of the software complexity and thus the need for strongly structured programming methodology, such as skill-based programming, and b) doubts and mistrust of the general stability of software solutions in comparison to mechanical approaches.

*Lorenz Halt and Philipp Tenbrock are with the Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Stuttgart, Germany {lth,pgt}@ipa.fhg.de

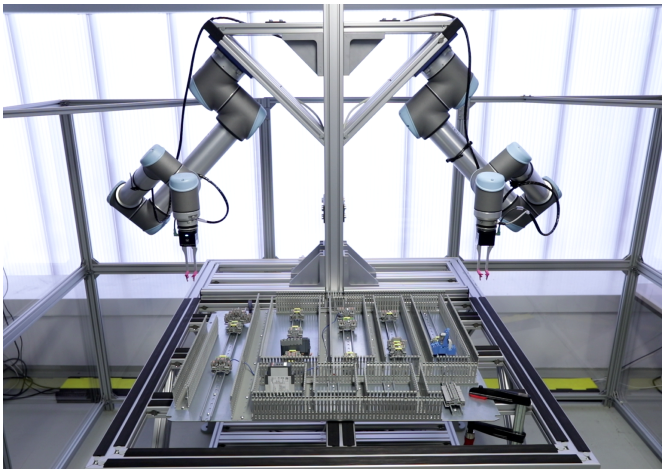

Fig. 1. Robotic dual arm setup for automated wiring of electrical cabinets[2].

Inspired by human anatomy, two Universal Robot UR10 arms were installed in a *shoulder-like* configuration (Fig. 1). Both were equipped with a force-torque sensor and a two-finger electrical gripper, each. The gripper fingers serve two distinct purposes — fixing the to-be-connected tip of the wire and guiding the wire while routing through the cable ducts. The fingers were specially designed for being a simple mass product without active components.

The robotic production cell is loaded manually with a fully equipped base plate including all electrical components and a feeder of prefabricated wires.

II. Methods and Technology

During runtime, the task specification formalism *iTaSC* [1], [2] is utilized to describe and solve the control problem based on the active skills of the composed application. *iTaSC* uses closed kinematic loops to map task-specific control problems expressed in their specific feature space (e.g. force control represented in a Cartesian force frame) to an arbitrary kinematic chain (i.e. target joint velocities of an industrial robot).

For multi-arm applications one loop may contain several robots, effectively concatenating them to a single manipulator with one shared feature space or involve multiple closed kinematic loops each containing separate robots and respective feature spaces.

By realizing an iTaSC solver with the task-priority strategy [3], [4], simultaneous control problems can be handled within respective nullspace of higher prioritized constraints.

[2]https://youtu.be/xXR2FxPVqa4

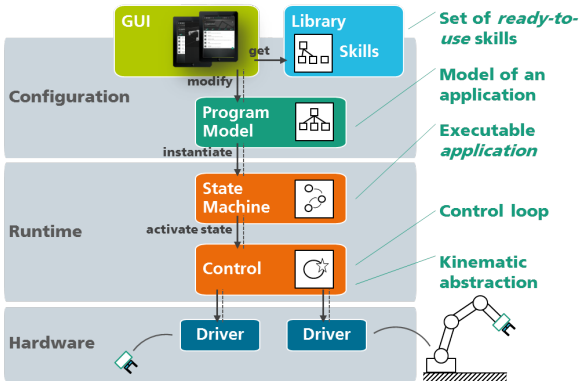

Fig. 2. Schematic of the control framework

New measurement data, i.e. robot joint positions and external force-torque-sensor measurements, is acquired continuously and the kinematic loop equations of *iTaSC* are solved in real-time. In this way, reference velocities for the robot joints are calculated accordingly and commanded to the robot.

The application is composed of building blocks named *skills*. All skills have a similar appearance, despite modeling different behaviors. Some being purely compositional to organize the program flow, others reflecting elementary robotic capabilities or represent complete composites of e.g. assembly strategies. An overview of general and elementary skills can be found in [5], [6].

Every skill is ultimately based on elementary skills that encapsulate basic robot behavior, e.g. a linear movement. Composite skills may include elementary skills directly or nested sub-composites to define complex behaviors — e.g. a parallel hierarchy of nested movement and force skills. Other possibilities include sequences of skills and the arrangement as general state machines. Every skill transition is guarded by one or more *monitors*. Skills may come with predefined monitors. The target of a respective transition may be inherited from a higher-level compositional skill or is specified explicitly. Further, a skill may overwrite its inherited *controller* with a customized special-purpose implementation [7].

The application and all compositional elements are represented and prototyped using an *XML* based DSL — in particular skills, monitors and controllers [8]. Reference velocities for the robot joints are calculated based on the runtime loop described before. This leads to immediate reusability of skills and subskills, as well as skill components, such as *monitors* and *controllers*. The overall control framework is depicted in Fig. 2.

III. Application and Development

For the automated wiring application several challenging subtasks were resolved. Based on the extensive existing skill library no elementary behavior needed to be implemented. Project-specific implementations were limited only to specialized force controllers — able to cope with single-sided nonlinear behaviour of *rope/wire pulling*. The main work of the application development consists of skill composition and orchestration.

The application was divided into subtasks, represented by macro-skills respectively (Fig. 3). The application is loaded with a set of parameters extracted from the wiring plan represented in CAD. The usual coordination mechanism of the control framework consists of a hierarchical state-chart that is statically created from the application's skills in a systematic way, prior to execution. However, as the properties and routing information of the wires differ to a large extent, the macro-skills have to be reparametrized and rescheduled during runtime. This was modelled by an additional *Coordinator* entity that stores all parameters and prepares a specifically parametrized macro-skill sequence. Besides the parametrization of, e.g. push-in poses and the wiring route waypoints, the most significant decision based on the parameters is the direction of the wiring and thus which robot will perform the first push-in and the primary routing. This decision is made based on heuristics respecting the reachability of the robots and the danger of entangling. The *Coordinator* technically is a skill that does not include any motion but executes the parametrized macro-skills. As the dynamic reconfiguration of the macro-skills is a non-typical requirement, the *Coordinator* was hand-crafted for this application and does not follow a generalized model.

The first subtask (A) picks up the wire and moves to an initial pose, the next task (B) moves the wire to the first push-in position. Consequently, task (C) performs the push-in and releases the wire. Regrasping of the wire is realized in (D) for switching the function of the first gripper from fixing the cable to guiding. This task further includes the initial diving into the cable duct. The routing of the wire may consist of several straight lines with intermediate 90° corners. Task (E) models one straight motion within the cable duct and task (F) models rotation around a corner. Most routes will include several executions of differently parametrized tasks (E) and (F). The last part of the route is modeled in (G) separately. Finally, (C) performs the second push-in. Task (H) is executed with both, the first and the second arm, to perform a delicate operation to push the wires into the fixation of the cable ducts at the entry and exit locations of the wire. For completeness, a recovery and homing task (I) is included.

The advantages of the constraint-based programming approach particularly emerged for the subtasks (B) and (E). During subtask (B), the two robots rotate downwards in a coordinated movement. The operation is defined as a hierarchy of two concurrent subtasks: 1) The wrist joints rotation is defined in joint space, while 2) the horizontal position of the robots' TCPs is kept constant. This is necessary to avoid both the wire being wrapped around the tools and the wire being stretched. During subtask (E), the first robot moves along the cable duct, routing the wire, while the second robot moves down to avoid stretching the wire. Through the constraint-based task specification approach, this was possible in an elegant and targeted manner. The height of the robot holding

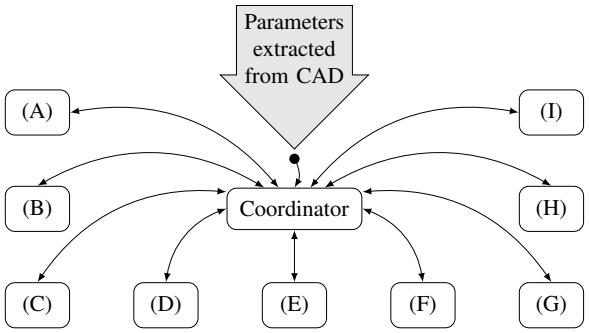

Fig. 3. High level overview of the composed state machine. Each shown *macroskill* is parametrized by the *Coordinator* during runtime and contains multiple levels of nested sub-state machines.

the wire is defined by the travel path of the routing robot. In contrast, native robot programming approaches typically focus on individual robots, while coordination is usually introduced through synchronization points in the robot program.

Additionally, the constraint-based programming approach made manual transformations between the task space of the previously planned wiring map and the robot operation space obsolete. To achieve this, the task space coordinate frame was placed in the corner of the cabinet's base plate, with the axes aligned to the plate. Thus, the CAD data could be used as target coordinates without preprocessing, while the transformation to robot operation space was handled by the *iTaSC* formalism.

A *divide-and-conquer* approach with a strong skill-based programming scheme was essential for realizing this project. Each macro-skill was individually designed, composed, and tested. Rigorous separation of concerns paved the way for efficient parallel developments. A high reuse rate of previously developed and thoroughly tested skills minimized the risk of software bugs. The constraint-based programming approach made manual transformations between the bird's-eye view task space of the planned wiring map and the robot operational space obsolete and reduced the overall development time dramatically.

REFERENCES

[1] J. D. Schutter, T. D. 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, 2007. [Online]. Available: http://ijr.sagepub.com/content/26/5/433.full.pdf

[2] R. Smits, T. D. Laet, K. Claes, H. Bruyninckx, and J. D. Schutter, "iTASC: a tool for multi-sensor integration in robot manipulation," in *2008 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*. Piscataway, NJ: IEEE Computer Society, 2008, pp. 426–433.

[3] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority Based Redundancy Control of Robot Manipulators," vol. 6, no. 2, pp. 3–15, 1987.

[4] B. Siciliano and J.-J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *1991 Int. Conf. on Advanced Robotics*, 1991, pp. 1211–1216 vol.2.

[5] L. Halt, F. Nägele, P. Tenbrock, and A. Pott, "Intuitive constraint-based robot programming for robotic assembly tasks," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE Computer Society, 2018.

[6] L. Halt, P. Tenbrock, F. Nägele, and A. Pott, "On the implementation of transferable assembly applications for industrial robots," in *2018 IEEE . Symposium on Robotics (ISR)*. VDE Verlag, 2018.

[7] F. Nägele, L. Halt, P. Tenbrock, and A. Pott, "Composition and Incremental Refinement of Skill Models for Robotic Assembly Tasks," in *2019 IEEE Int. Conf. on Robotic Computing (IRC)*. Los Alamitos: Conference Publishing Services, IEEE Computer Society, 2019, pp. 177–182.

[8] ——, "A prototype-based skill model for specifying robotic assembly tasks," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE Computer Society, 2018.