

On the Programming and System Integration of
Robots in Flexible Manufacturing

On the Programming and
System Integration of
Robots in Flexible Manufacturing

Mathias Haage

Department of Computer Science
Lund University
Lund, December 2010

Department of Computer Science
Lund University
Box 118
SE-221 00 LUND
Sweden

ISSN 0280-5316
ISRN LUTFD2/TFRT--XXXX--SE

© 2010 by Mathias Haage. All rights reserved.
Printed in Sweden by XXXX.
Lund 2010

Abstract

Advanced manufacturing technologies, and programmable machines such as industrial robots, are used to increase productivity and quality for competitiveness on a global market. Development of increasingly flexible manufacturing systems has resulted in an increasing importance of software aspects, both on a system level and for efficient interaction with human operators. Trends toward providing customized products increase the need for flexibility, which implies a need to build modular systems that are flexible enough to handle frequent changes in production operations and product designs.

The objective of the research presented in this thesis is to improve the flexibility of industrial robot software when used as a component in flexible and reconfigurable industrial automation solutions. Contributions are made in four areas; First, high performance industrial motion control is enhanced to utilize arbitrary sensors in task definition and execution. Results include an extensible task programming language, allowing for flexible integration of sensor motion in established robot languages. Second, flexibility of the robot structure itself is studied, with an emphasis on software tool configuration support for a highly modular parallel kinematic robot featuring stiff motions and large workspace. Third, several operator interaction techniques are evaluated for fast and easy robot setup. Novel interaction devices and use of sensors bring new opportunities to improve robot setup procedures. Finally, and also pointing out future research directions, semantic web techniques are explored for use within automatic generation of user interfaces from product and process data, and for more efficient integration of offline engineering tools in the workflow for online task generation.

The findings are based on a variety of industrial prototypes and case studies, with novel software solutions ranging from low-level device interfaces to high-level semantic integration. The experienced resulting enhancements of flexibility, usability and modularity are encouraging.

Preface

My interest in computers started when I was nine years old and got my first computer. Actually it belonged to my father, but I soon adopted it for myself. It was the beginning of the 80s when the personal computer first appeared and was offered at reasonable prices in shops. There was no networking, almost no available software, and programming was performed on built-in interpreters/editors. My first computer, a Sinclair ZX Spectrum, taught me to write programs very quickly through its excellent programming interface. It was especially interesting to try to create graphics demos, even though the competitor, the Commodore 64 was a better performer. Since this moment, programmable entities have always fascinated me.

I did not get into contact with robotics until much later, after being recruited for teaching computer graphics at the department. Here I came in contact with robotics research for the first time. I discovered that programmable machines were at least as fascinating as computers. Of particular interest were the human-robot interfaces, since it was painstakingly time-consuming to produce robot programs. This opened up my interest to the human and engineering aspects of robotics, which I have tried to pursue during the years I have worked with the thesis.

Applied industrial robotics is a very experimental field, driven forward by prototypes performing in industrial contexts. Prototype development is usually a large team effort with many engineering hours spent and many parties involved before any results are visible. A particular challenge has been to extract and identify scientific issues and contributions within the torrent of engineering work surrounding prototypes. I have done my best to this end. Looking back at the chosen path as it turned out, I am quite happy with what I have done.

Acknowledgement

This thesis would not have come into being without the support and encouragement of a great many people and partners. First of all I would like to thank my supervisor Klas Nilsson, who showed me the field and gave me the opportunity to embark on this endeavor. His support and guidance over the years has been invaluable. I would also like to thank my co-supervisors, who have all given me support at some point during the journey, Görel Hedin, Martin Nilsson, and Lennart Olsson. I would like to thank Jan Bohman for employing me at the department, where I first came in contact with robotics research.

The people centered around the robot lab has played a large part in my thesis, whether as co-authors of papers, helping with implementations, or giving advice. From my department I would like to mention Pierre Nugues, Jacek Malec, Sven Gestegårdh-Robertz, and Anders Nilsson. From the department of Automatic Control I would like to mention my co-workers Anders Robertsson, Rolf Johansson, Anders Blomdell, Tomas Olsson, Johan Bengtsson, and Isolde Dressler. I also extend my gratitude towards the people at the mathematics department that I have been fortunate to meet and work with. Gunnar Bolmsjö, at the division of Machine Design, inspired me in part of the work.

The department staff has been very helpful in all aspects of my work. I thank you all. I would especially like to mention Lars Nilsson and Radosław Szymanek for their support and many interesting lunch conversations. I would also like to acknowledge the people behind the software frameworks JastAdd and JaCoP, that I have used in a few places of the thesis; Görel Hedin, Torbjörn Ekman, Radosław Szymanek, and Krzysztof Kuchcinski. I would like to thank the numerous Master theses students and project workers that I have worked with during the thesis. A special thanks to Johan Lauri for willingly posing as the operator in the thesis overview picture.

During my work I also had the opportunity to work with people from several academic departments outside Lund University. I would especially like to mention Gilbert Ossbahr, Henrik Kihlman, and Mats Björkman from the department of Management and Engineering, division of Assembly Technology, at Linköping University. I extend my gratitude to all the people I have worked with at the Fraunhofer Institute for Manufacturing Engineering and Automation during projects. I would like to mention Matthias Bengel and Marius Pflüger. I would also like extend my gratitude to the people I met from the Industrial Robotics Laboratory in the Mechanical Engineering Department, University of Coimbra, headed by Norberto Pires.

Preface

I have had the opportunity to work with several companies during the projects.

- I have worked with people from several ABB companies during the years, when working with their industrial control systems and software tools; Robotics, Corporate Research, Flexible Automation, and Automation systems. I especially want to mention Torgny Brogårdh, Håkan Brantmark, Peter Eriksson, Jens Hofschulte, and Magnus K. Gustafsson.
- Kranendonk in Tiel, Holland and KPS Rinas in Køge, Denmark have been very helpful and supporting when working with industrial application scenarios and demonstrators. I want to especially thank Svend Søhald, Magnus Kjærbo, and Aleksandar Milojevic.
- I want to thank the people at Visual Components Oy in Finland for access to and help with their software. I want to mention Juha Renfors and Ricardo Velez.
- The Jayway company located in Malmö, Sweden, was very helpful when I integrated the Anoto digital paper technology into application prototypes. I extend my gratitude to all co-workers there.
- I want to thank Saab Aerostructures and Magnus Engström for their input on challenging application needs in aerostructure drilling.
- I am also grateful towards Güdel AG in Langenthal, Switzerland, for their investments and efforts with building the prototype parallel-kinematic manipulator.
- Finally, I want to extend my gratitude to all other system integrators and manufacturing companies that I have met in brief moments over the years, for instance at tradefairs, and application tests.

Our work has been funded from several national and European agencies. I want to acknowledge the EU 5th Framework Growth Project for work within the project *affordable, flexible system for off-line automated fettling and finishing* (AUTOFETT). The EU 6th Framework Growth Project for work within the project *the European robot initiative for strengthening the competitiveness of SMEs in manufacturing* (SMERobot) and the project *skill-based inspection and assembly for reconfigurable automation systems* (SIARAS). NUTEK (The Swedish National Board for Industrial and Technical Development) for work within Complex Technical Systems, Open Control Architectures. SSF (Swedish Foundation for Strategic Research) for work within the project *flexible and accurate manufacturing operations using robot systems* (FlexAA).

Dedication

To Kasia

Contents

1. Introduction	1
1.1 Outline and Contributions	4
Part I. Sensor-based Robot Motions	13
2. Variable Time Delays in Visual Servoing and Task Execution Control	15
2.1 Introduction	15
2.2 Delay and Noise in Feedback Control Loop	16
2.3 Pursuit-and-Capture Experiment	19
2.4 Results	23
2.5 Discussion	25
2.6 Conclusions	27
3. Extending an Industrial Robot Controller	29
3.1 Introduction	29
3.2 Considerations and Design of System Extensions	32
3.3 High Power Stub Grinding	41
3.4 Discussion	46
3.5 Conclusions	48
4. Case Study: Cost-Efficient Drilling	49
4.1 Introduction	49
4.2 System Topology	52
4.3 Drilling System Design	56
4.4 Experiments	60
4.5 Discussion	62
4.6 Conclusions	64
Part II. Configurable Modular Equipment	67
5. Configuration Support for Parallel Manipulators	69
5.1 Introduction	69
5.2 Gantry-Tau Kinematics	72

5.3	Gantry-Tau Configuration Support	77
5.4	Conclusions	82
Part III. Human-Robot Interfaces		85
6.	On the scalability of visualization in manufacturing . .	87
6.1	Introduction	87
6.2	Related work	88
6.3	Scalability	89
6.4	Executable visualization	90
6.5	Implementation	93
6.6	Applications and discussion	96
6.7	Conclusions	99
7.	A Robot Speech Interface with Multimodal Feedback .	101
7.1	Introduction	101
7.2	Multimodal interfaces and robot programming tools . .	102
7.3	Prototype	104
7.4	Ongoing experiments and future work	111
7.5	Conclusion	112
8.	Human-Robot Interaction using Advances Devices . . .	113
8.1	Introduction	113
8.2	The Anoto Digital Pen	114
8.3	Development Towards Foundry Applications	117
8.4	Conclusion	120
Part IV. Semantic Robot Interfaces		121
9.	Toward Ontologies and Services for Assisting Industrial Robot Setup and Instruction	123
9.1	Introduction	123
9.2	Multimodal Form-based Dialogue	124
9.3	Ontology Modeling	125
9.4	Prototype Setup	128
9.5	Conclusions	137
10.	Service Oriented Architecture for Automatic Planning and Programming of Industrial Robots	139
10.1	Introduction	139
10.2	Experimental Platform	141
10.3	Deployment of Programs	144
10.4	System Integration and Workflow Support	147
10.5	OWL-S Extension for Constraint-Based Search	152
10.6	Results	160
10.7	Conclusions	161
11.	Bibliography	163

1

Introduction

Industrial robots in manufacturing have traditionally been seen as general-purpose positioning devices, shaped by automation needs in long-batch production lines for several decades. They excel with high performance and consistent high quality in a number of industrial tasks such as welding, painting, and material handling. They are typically used in well-known environments with little or no variability in task and workpieces. Commissioning is a long process involving many hours of planning, simulation, programming, configuration, testing, and tuning, to make sure the robot will perform as expected when installation is completed.

Today the traditional view of the manufacturing robot is changing; the robot is becoming a more versatile tool. New emerging automation stakeholders promote the robot as a workshop tool that the operator can use to perform tasks. Shorter batches require robots that quickly and frequently can be re-commissioned. Less structured environments demand resilience toward variations in environments, tasks, and products. Sensors are increasingly relied upon to cope with those variations. Much research is currently devoted to re-molding the traditional industrial robot into a more “intelligent” and flexible tool for the future. The challenges for moving forward today lies in the robot software.

A central theme in this thesis is the notion of “skilled motion”. Most of the projects I have worked with is approaching this concept in one way or another. The concept attempts to cover a paradigm shift that is currently happening in the expression of industrial motions. Knowledgeable motion primitives incorporate information about the environment to adjust and solve tasks on-the-fly, providing a much better flexibility and motion abstraction level than before. However, flexibility comes at the cost of complexity, and the skilled motion is far more complex than earlier primitive motions. Throughout the presented research, the support for skilled motion in tools and methods has been a central problem. This is crucial to study, if robots are ever going to be simple to use and productive.

The thesis is divided into four parts, each with it’s topics and contributions, as shown in Figure 1.1.

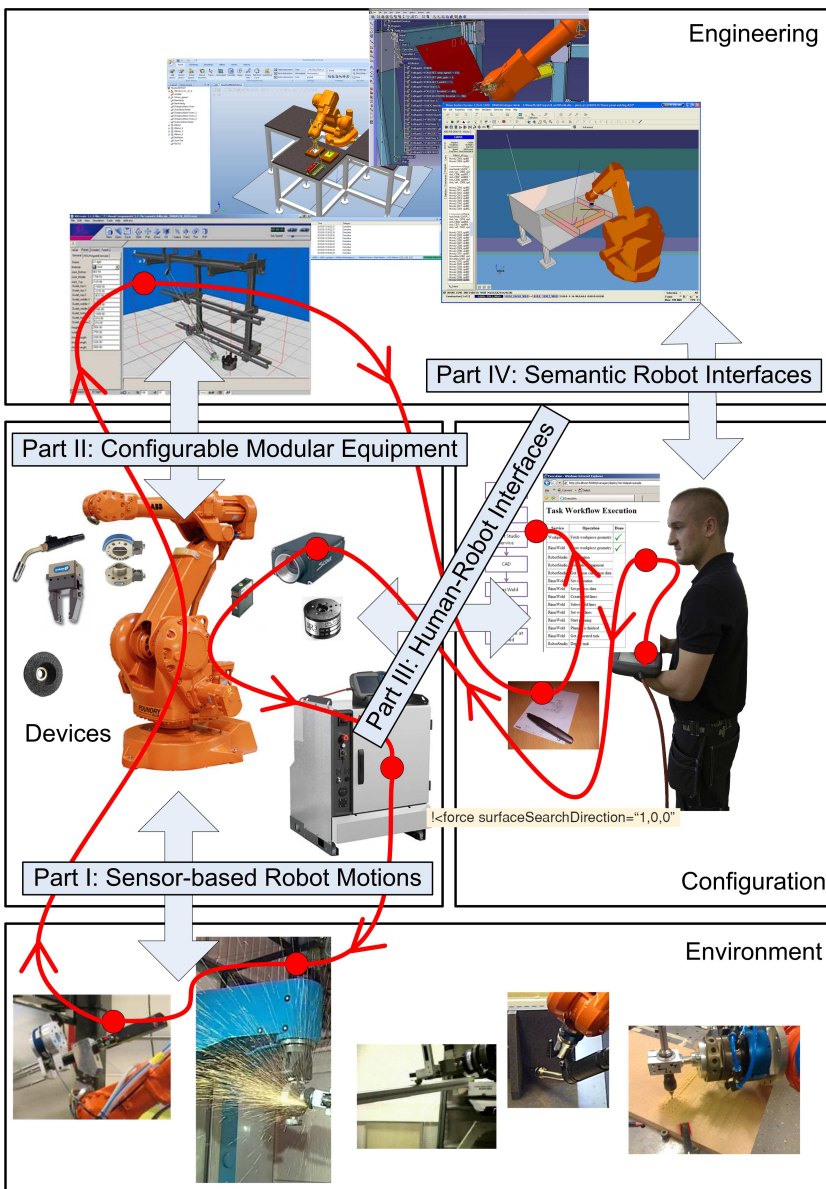


Figure 1.1 Four parts comprise the thesis. The thread shows the chronological order of publications.

- I **Sensor-based robot motions** need system-level support for implementation. The first part of the thesis starts with developing a vision-based sensor-feedback motion on a research controller system. This is followed by presenting a sensor interface for “fast” sensor input in an ABB robot controller. The sensor influence is specified graphically using an engineering tool and supported at the system-level programming language. The interface is evaluated by industrial applications in low-value (foundry) and high-value (aero) product segments using force feedback to improve machining operations.
- II **Configurable modular equipment**, here exemplified with a support structure and a parallel-kinematic robot concept, offers powerful structural customization properties. The second part presents the modular support structure and robot concept, and continues to present methods to support the initial structural dimensioning and task analysis of the robot structure in engineering tools. The method is evaluated by supporting the development of an industrial application in the foundry industry.
- III **Human-robot interfaces** are crucial for configuring complex tasks and motions. This area is huge and the third part presents three dives into the field. The first dive allows limited CAD functionality in shop-floor operator interfaces to make configuration easier. Enabling technologies are investigated. The idea is further visited as part of the welding demonstrator described in part IV, where a configuration support tool uses simulation imagery to create custom shop-floor setup dialogues. The second dive uses small task domain vocabularies to provide domain-specific programming support. A voice activated robot jogging and programming interface is evaluated. Finally, in the last dive, digital paper is presented as a new human-robot interaction medium for programming robots via pen and paper; direct CAD-annotation and form-based configuration is evaluated.
- IV **Semantic robot interfaces** automate human-robot configuration and programming tasks. By increasing the level of machine-understandable information in the robot system, it may be possible to reduce both time for configuration, and lower the risk for introduction of faults during setup. The fourth part presents two experiments. The first utilizes semantic information to automatically synthesize robot configuration dialogues for several modalities, optionally executing in parallel. The second experiment automatically integrates and configures a task planner, cell configuration tool and product knowledge to deploy and execute a task.

1.1 Outline and Contributions

This section contains a brief outline of each part in the rest of the thesis, with a description of contents, contributions, and related publications. Included chapters are condensed, merged, and/or updated versions of the content of the published papers.

Part I: Sensor-based Robot Motions

Improving the connection to the environment is crucial for handling uncertainties and variations, and can even enable whole new ranges of tasks. Sensors have been included in robot cells for a long time, such as laser sensors for weld seam tracking and camera sensors for picking. But for certain classes of tasks the performance offered by current controllers is not enough, such as contact tasks in stiff environments. The question is how to cope with high-performance sensing in robot controllers so that flexibility, safety, and robustness is kept. This research has been governed by the following questions:

- Is it technically feasible from an embedded systems point of view to permit third parties to extend the real-time motion control of industrial (high-performance) robots?
- If so, how can such increased motion capabilities be incorporated in the robot task specification?
- If such systems prove to be valuable (e.g. by improved productivity or robustness) in important robot applications, how should then the engineering support look like for efficiently using the new sensor-based robot applications?

The approach described provides an unique blend between an industrially feasible solution and flexibility through access to low level control.

Publications

The first paper suggests a vision-based sensor-feedback motion on a research controller system. The next paper describes a sensor interface for “fast” sensor input to an ABB robot controller. The sensor influence is specified graphically using an engineering tool and supported by a system-level programming language. In the third paper a machining task within the foundry industry is automated using force feedback to evaluate the interface. The fourth paper evaluates the interface further by improving another machining operation for a product within a different (high-value) product segment.

- Bengtsson, J., Haage, M. and Johansson, R. 2002. "Variable Time Delays in Visual Servoing and Task Execution Control." *Mechatronic systems 2002: a proceedings volume from the 2nd IFAC Conference*.
- Blomdell, A., Bolmsjö, G., Brogårdh, T., Cederberg, P., Isaksson, M., Johansson, R., Haage, M., Nilsson, K., Olsson, M., Olsson, T., Robertsson, A., and Wang, J. J. 2005. "Extending an industrial robot controller." *IEEE Robotics and Automation Magazine*.
- Robertsson, A., Olsson, T., Johansson, R., Blomdell, A., Nilsson, K., Haage, M., Lauwers, B., and de Baerdemaeker, H. 2006. "Implementation of industrial robot force control case study: High power stub grinding and deburring." *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Olsson, T., Haage, M., Kihlman, H., Johansson, R., Nilsson, K., Robertsson, A., Björkman, M., Isaksson, R., Ossbahr, G., and Brogårdh, T. 2009. "Cost-efficient drilling using industrial robots with high-bandwidth force feedback." *Robotics and Computer-Integrated Manufacturing*.

The second and third paper have been condensed into one chapter.

Contributions

The author has mostly tackled the second and third question. In the first paper the author participated in development of the experiment and contributed with experimental design and the vision-based approach. In the second to fourth paper the author participated in the experiment design and development, and contributed with the system-level language extension that allowed sensor-based control to be integrated within the normal task specification. The fourth paper uses a language extension based on modular compiler-compiler tools allowing for a declarative description of the extension.

Findings

From the results presented in the papers, several conclusions were drawn:

- ◇ Properly designed (in terms of modularity, efficiency, and safety) control system interfaces, together with matching external control functions (including state machines and an appropriate interplay with the task execution), permits high-performance motion control to be extended via customer add-ons.
- ◇ Based on a modular view on robot languages, supported by a highly

modular compiler toolkit, the add-ons on the motion control level can be reflected on a user level in a powerful and convenient way that promotes engineering efficiency.

- ◇ From an engineering point of view, standard available software tools can be easily extended such that they apply on a motion-control level, and in the corresponding way standard CAM tools can be extended to connect the increased robot capabilities to CAD-based robot programming.

Each of these items, as well as their integration into complete systems according to the presented prototypes, represent contributions beyond existing solutions within sensor based robot motions. The authors of the fourth article received the 2004 EURON Technology Transfer Award.

Part II: Configurable Modular Equipment

Configurable modular equipment offers powerful customization properties. But engineering tool support is necessary to efficiently explore and make decisions in the available customization space. This part presents a modular support structure and robot concept, and continues to present engineering tool methods to explore the initial structural dimensioning and task analysis of the robot structure. This research has been governed by the following questions:

- Is it feasible (from mechanic, kinematic, and dynamic viewpoints) to utilize the modularity of subcomponents for parallel-kinematic manipulators for more configurable manufacturing systems?
- If so, can typical end-user requirements for setup, calibration and task programming be met?
- If such components prove to be useful (e.g. by flexibility or cost-effectiveness) in automation systems, how should engineering support look like for efficiently using robot subcomponents to develop automation applications?

The approach described provides a unique blend between industrially feasible modular support structures and novel robot concepts.

Publications

The first paper presents a parallel-kinematic manipulator that provides a stiff and large workspace, built from modular structural concepts. The robot is evaluated through several prototype builds, including an installation in a foundry workshop. Engineering tools are used for the initial

kinematic dimensioning of the robot for the foundry cell. The second paper presents results from several later prototype builds.

Dressler, I., Haage, M., Nilsson, K., Johansson, R., Robertsson, A., and Brogårdh, T. 2007. "Configuration support and kinematics for a reconfigurable gantry-tau manipulator." *Proceedings 2007 IEEE International Conference on Robotics and Automation*.

Haage, M., Dressler, I., Robertsson, A., Nilsson, K., Brogårdh, T., and Johansson, R. 2009. "Reconfigurable parallel kinematic manipulator for flexible manufacturing." *Preprints of the 13th IFAC Symposium on Information Control Problems in Manufacturing*.

The two papers have been condensed into one chapter.

Contributions

The author participated in earlier prototype builds of the table-sized manipulators. The author contributed in the early planning phase of the foundry prototype with development of methods allowing for task-dependent dimensioning and structure-invariant task programming of the robot structure in simulation tools.

Findings

The conclusions drawn are:

- ◇ Prototype development of a new parallel-kinematic structure (called Gantry-Tau) combined with a new modular support structure (called BoxJoint) demonstrate new opportunities for the utilization of parallel robots in automation toolboxes.
- ◇ Engineering tool support by means of properly parameterized template models (in terms of support and robot modules) offering kinematic structure-invariant programming features can efficiently support early structural design decisions.

Each of these items, and their integration into complete systems according to the presented prototypes, represent novel contributions within modular industrial robotics.

Part III: Human-Robot Interfaces

Human-robot interfaces are vital for configuring complex tasks and motions in order to minimize setup time and reduce errors in the robot cell.

As the trend moves toward more frequent operator interaction, new methods and devices for robot setup and programming need to be evaluated. The research presented has been driven by the following questions:

- What classes (depending on input data type, involved parties, sensors, and devices) of user interfaces will be important in future robot application scenarios?
- What classes of user interface technologies are feasible to provide functionality for robot operators given available devices, sensors, and software tools?
- Are there any best solutions that are applicable to many use cases?

Publications

The area of human-robot interaction is huge and the third part only presents three dives into the field. The first paper investigates visualization platforms that could allow limited CAD functionality in shop-floor operator interfaces. An evaluation of visualization technologies are presented (current at paper publishing date). In part four this is exemplified in a configuration support tool that uses simulation imagery to create customized shop-floor setup dialogues. The second paper uses small task domain vocabularies to provide domain-specific programming support. Voice is used as a complimentary input channel configurable for specific tasks. A voice activated robot jogging and programming interface is evaluated. The third paper evaluates digital paper as a new human-robot interaction device. Direct annotation of CAD drawings is evaluated. The first paper in part four also evaluates form-based configuration for configuration of door signs in a wood milling cell.

Haage, M. and Nilsson, K. 1999. "On the scalability of visualization in manufacturing." *7th IEEE International Conference on Emerging Technologies and Factory Automation*.

Haage, M., Schötz, S., and Nugues, P. 2002. "A prototype speech robotic interface with multimodal feedback." *11th IEEE International Workshop on Robot and Human Interactive Communication*.

Pires, N. J., Godinho, T., Nilsson, K., Haage, M., and Meyer, C. 2007. "Programming industrial robots using advanced input-output devices: Test-case example using a CAD package and a digital pen based on the Anoto technology." *International Journal of Online Engineering*.

Contributions

In the first paper the author contributed with ideas and to prototype implementation. The idea is revisited in the patent-pending deployment wizard described in part four. For the second paper the author contributed with idea and experimental setup. In the third paper the author contributed with application analysis and experimental setup for the direct CAD input and form-based input.

Findings

Three classes of user interfaces were evaluated in prototype instances; complimentary and redundant interfaces, interfaces using data and/or functionality from engineering tools, and configuration interfaces. This led to the following findings:

- ◇ Each evaluation instance indicate niche potential, but it is hard to draw any conclusions regarding best solutions. This indicates an open architecture for handling many input devices and input methods in a robust fashion appears to be the the most suitable approach for future systems and research.
- ◇ The digital paper works well as a modality for form-based input. The evaluation prototypes show that configuration interfaces are useful for task programming and cell re-configuration, given an appropriate support architecture for managing and exposing configuration options.

The interface technologies presented, together with their integration into prototype systems, represent contributions toward finding new interface technologies supporting future robot application scenarios.

Part IV: Semantic Robot Interfaces

Semantic robot interfaces aim at assisting the operator in configuring and programming complex robot tasks. This is done by increasing the level of machine-understandable knowledge about involved entities such as product, process, equipment and planning tools, thereby enabling reasoning mechanisms to provide the desired support. The research has been driven by the following questions:

- Is it technically feasible to automate some of the engineering operations that normally require human intervention? If so, what information structures and architectures are needed?

- Can typical end-user tasks for setup and task programming be significantly improved (time spent and error reduction)? What and where are the limitations?

The two experiments described present novel approaches toward industrially feasible setup and programming architectures for industrial robots.

Publications

The two experiments are presented in one paper each. The first paper automate generation of system-initiative user interface dialogues from product and process knowledge. The evaluation prototype synthesizes robot configuration dialogues for several modalities, including digital paper, voice, and web interfaces. The second paper presents a semantic service architecture for automatic integration and invocation of task planners and other CAD software with product, process and cell knowledge. The evaluation prototype performs goal-based task planning of a welding task, simplifying use and configuration of the involved engineering tools.

Haage, M., Nilsson, A., and Nugues, P. (2008). "Toward ontologies and services for assisting industrial robot setup and instruction." *5th International Conference on Informatics in Control, Automation and Robotics*.

Bolmsjö, G., Haage, M., Søhald, S., Kjærbo, M., and Gustafsson, M. K. (2010). "Service Oriented Architecture for Automatic Planning and Programming of Industrial Robots." *20th International Conference on Flexible Automation and Intelligent Manufacturing*.

Contributions

In the first paper the author participated in the application analysis and in performing the experiments. In the second paper the author provided the idea, participated in the experimental design and setup of the robot test bed incorporating a commercial welding task planner.

Findings

Whereas the two experiments indicate good results in small contexts, further investigation is needed concerning the applicability in general. Development of common bodies of machine-understandable knowledge within the manufacturing domain is then necessary; without available knowledge, reasoning techniques do not apply. More specifically, of value for further research, it can be concluded that:

- ◇ Methodology for capturing and incorporating manufacturing concepts into working knowledge bodies need to be developed such that information structures (e.g. in terms of ontologies) can be inferred gradually. For instance, even more or less standardized information entities such as geometries for work pieces, need to embody context-specific requirements from third parties, relating the geometric parts to the intended manufacturing requirements. Rather than the more fixed approaches provided by CAD systems, these requirements need to be captured and incorporated into a working body of knowledge including semantic information.
- ◇ Production engineering is often rather complex, with workflows involving a variety of software tools and equipment, such as sensors and end-effectors, that need to be selected, configured, and used appropriately. The existing workflows are not really suited for use in small manufacturing facilities where it is not feasible to have the needed software/equipment specialists. This hampers wide-spread use of robots as targeted by the SMErobot initiative. Useful semantic models, for well-defined domains such as arc welding or small parts assembly, would allow engineering knowledge to be stored for reuse and then applied (semi-)automatically across involved tools and equipment. As the prototype implementation shows, reasoning then fits as dedicated components that can ease the configuration and robot programming. Thereby use of robots directly on the shop-floor, by non-expert operators, appears to be tractable.

These items can also be interpreted in terms of scalability, with improvements downwards to small-scale usage of robots, and upwards to more complex application requiring skilled (sensor-based) robot motions. The findings in all the other parts together, are believed to contribute to the applicability of robots in present and future industrial applications.

Part I

**Sensor-based Robot
Motions**

2

Variable Time Delays in Visual Servoing and Task Execution Control

Reaction from visual data over existing networks and in presence of occlusion

2.1 Introduction

Network delays arises when closing vision feedback loops over non-dedicated networks with non-deterministic computational nodes. To build on existing networking, our system is implemented on a TCP/UDP intranet with vision and control processing performed on a workstation and a PC running soft realtime operating systems. Such setups give rise to communication and computational time delays. Usually the delays are varying to a degree which give rise to degraded control robustness in the system with resulting lack of performance or even instability.

Another problem is noise introduced into the system due to erroneous information and loss of information. For instance, a source of erroneous information is image measurement errors and a source of information loss is occlusion. Noise may result in significant loss of system performance.

We examine an image plane linear prediction strategy for compensating for delays and noise in an image based visual servoing loop (IBVS). The IBVS loop in consideration is illustrated in Fig. 2.1. The purpose of IBVS is to use uncalibrated cameras to control image plane motion [1]. IBVS loops are typically used for positioning tasks and the use of uncalibrated cameras make the IBVS loop attractive in an industrial context.

The problems considered in the studied visual servoing loop are:

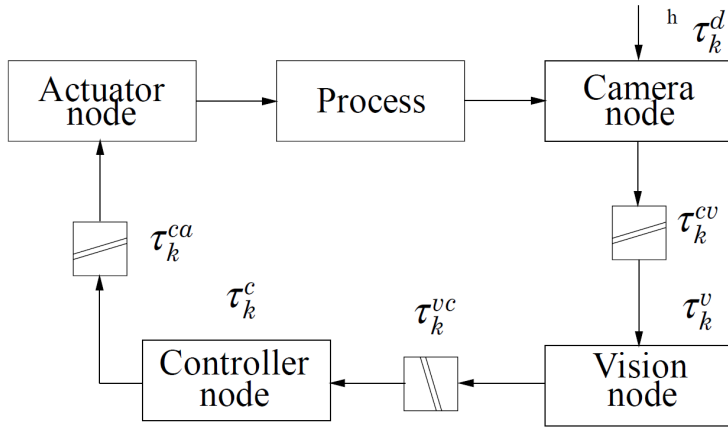


Figure 2.1 Distributed digital control system with induced delays, τ_k^{cv} , τ_k^{vc} and τ_k^{ca} . The computational delay in the vision node is τ_k^v , the controller node τ_k^c and the digital camera node τ_k^d .

- Presence of variable time delays arising from network communication and image processing in one computational node.
- Asynchronous data flow from two digital cameras giving rise to unknown time delays.
- Noisy data from extraction of object feature points in the camera image space.
- Losing track of the object. The image analysis algorithms start sending the wrong data. Typically we have lost track and the feature point has “jumped” to another object.
- The object is missing in the image, typically due to occlusion.

In this paper, we implement image plane linear prediction using a Kalman filter with nonuniform updating intervals. We verify experimentally that errors caused by variable time delays and noise can be sufficiently compensated for by satisfactorily performing an object pursuit-and-capture experiment. Prediction errors are measured and presented.

2.2 Delay and Noise in Feedback Control Loop

In order to compensate for time delays, occlusion and failed tracking, a dynamic model of the object of pursuit is used. The control is performed

in image space. Hence, we need to identify the dynamics of the object in the image space. The movement of the object in the work-space may be linear, but the movement in the image-plane will be nonlinear since the transformation between Cartesian space and image space is nonlinear. Even so, a linear model combined with a Kalman filter can be used for our purposes, since the experimental result shows that the upper bound on the nonlinearity is small.

In order to identify the dynamics, we use timestamped data. The controller runs with a fixed sampling rate and we want to identify the dynamics of the object in image-space with the same sampling rate. Therefore the feature points are resampled to the sampling rate of the controller.

$$\begin{aligned}
 \hat{x}_{\text{image}}^{\text{robot}}(k) &\rightarrow \tilde{x}_{\text{image}}^{\text{robot}}(k) \\
 \hat{y}_{\text{image}}^{\text{robot}}(k) &\rightarrow \tilde{y}_{\text{image}}^{\text{robot}}(k) \\
 \hat{x}_{\text{image}}^{\text{object}}(k) &\rightarrow \tilde{x}_{\text{image}}^{\text{object}}(k) \\
 \hat{y}_{\text{image}}^{\text{object}}(k) &\rightarrow \tilde{y}_{\text{image}}^{\text{object}}(k)
 \end{aligned} \tag{2.1}$$

where $\{\hat{x}, \hat{y}\}$ is a time-varying step, $\{\tilde{x}, \tilde{y}\}$ is the corresponding position after the resampling and k is the sample index.

Identification method A linear discrete-time time invariant system in state-space realization is used. The innovation model is

$$\begin{aligned}
 x_{k+1} &= Ax_k + Bu_k + Kw_k \\
 y_k &= Cx_k + Du_k + w_k
 \end{aligned} \tag{2.2}$$

where w_k is noise. An object undergoing linear movement with constant acceleration can be estimated as a second order state-space model.

Assuming the experimental data from the vision system is delayed less than one sample, it is enough to use a one-step Kalman filter predictor. The one-step predictor is given by:

$$\begin{aligned}
 \hat{x}_{k+1} &= A\hat{x}_k + Bu_k + K(y_k - \hat{y}_k) \\
 \hat{y}_k &= C\hat{x}_k + Du_k
 \end{aligned} \tag{2.3}$$

In order to use this Kalman filter based on fixed sampling time, the resampled data is used.

Nonuniform updating of Kalman filter Tracking and extraction of feature points of an object in presence of delays and noise is a difficult task.

Prediction using a dynamic object model and a Kalman filter could be used to improve tracking. In cases when we loose track and are following spurious objects, a fixed updating Kalman filter is not sufficient. We therefore propose that the Kalman filter should be updated only when a new measurement is close to the result from the model running in open loop. Otherwise, we use the prediction of the model running in open loop.

Estimation of prediction error

In our application context, we know that the object performs a linear motion and can therefore calculate the prediction by estimating the real trajectory in image space. This can be done by fitting a physical model of the object trajectory in world space to our measured data. In order to do this task, it is necessary to use calibrated cameras.

For a stereo rig with two asynchronously working cameras, it is necessary to resample the obtained data to a common sampling period in order to use triangulation for depth estimation.

$$(\hat{x}_1, \hat{t}_1)(\hat{x}_2, \hat{t}_2) \rightarrow (\tilde{x}_1, \tilde{x}_2, \tilde{t}) \quad (2.4)$$

where \hat{x}_1, \hat{x}_2 are the sampled image space coordinates from Cameras 1 and 2, \hat{t}_1, \hat{t}_2 are the corresponding sampling times, \tilde{x}_1, \tilde{x}_2 are the resampled image space coordinates from camera 1 and 2, and \tilde{t} is the corresponding resampling time.

3D reconstruction is done by using triangulation:

$$\bar{\lambda}_i \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} X_i \quad (2.5)$$

where $\bar{\lambda}_i$ is the scale factor for each sample, \tilde{x}_1, \tilde{x}_2 are the corresponding image coordinates, P_1, P_2 are the camera matrices for cameras 1 and 2, respectively, and X_i is the vector of world coordinates for the sample.

Using Eq. 2.5, world coordinates and scale factors are calculated for the trajectory. Assuming linear movement with constant acceleration, a trajectory is identified:

$$\begin{bmatrix} C \\ c \end{bmatrix} \begin{bmatrix} \tilde{t}^2 \\ \tilde{t} \\ 1 \end{bmatrix}_i = \begin{bmatrix} X \\ \bar{\lambda} \end{bmatrix}_i \quad (2.6)$$

$$\lambda_i = c \begin{bmatrix} \tilde{t}^2 \\ \tilde{t} \\ 1 \end{bmatrix}_i \quad (2.7)$$

$$(x_k)_i = P_k C \begin{bmatrix} \tilde{t}^2 \\ \tilde{t} \\ 1 \end{bmatrix}_i \lambda_i^{-1} \quad (2.8)$$

where $C \in \mathbb{R}^{4 \times 3}$ contains the coefficients of the trajectory, $c \in \mathbb{R}^{1 \times 3}$ contains the coefficients for a linearization of the scale factor, λ_i is the linearized scale factor for each sample, $(x_k)_i$ are the estimated real image coordinates for each sample, $k \in \{1, 2\}$ is the camera and i is the image sample. The prediction error is:

$$(e_k)_i = (x_k)_i - (\hat{x}_k)_i \quad (2.9)$$

2.3 Pursuit-and-Capture Experiment

The experimental setup is illustrated in Fig. 2.2. A stereo rig is looking down onto a ramp in an end-closed-loop configuration¹. The robot task is to pursue and catch an object moving along the ramp using visual feedback. Visual servoing is made both in the depth direction and in the image plane. Figures 2.3, 2.4, 2.5 and 2.6 illustrate a typical experiment run. The experimental setup has been designed to avoid occlusion and uses tracking to locate feature points.

Hardware An ABB Irb 2000 industrial 6-DOF robot is used as manipulator. The robot is configured to accept pregenerated trajectories as move commands. As for end effectors, the robot is equipped with a grip tool with variable grip pressure.

Two off-the-shelf Sony DFW-v300 digital cameras are mounted in a stereo rig configuration. The cameras deliver 320x240 resolution images at 30Hz in a 16 bit YUV422 color format. The images are sent asynchronously through an IEEE-1394 (FireWire) network.

A 450MHz PC is used as receiver for camera images and for performing image processing. Feature points from the image processing are sent through TCP/IP to a Sun workstation.

The visual servoing controller is run on a 2x450MHz Sun Ultra 60 workstation with the controller implemented as a Simulink model in Matlab. The controller sends pregenerated trajectory move commands to an embedded PowerPC controller board which executes a customized trajectory receiver in an open robot controller – see [2].

Vision Images received from each camera are processed to find specific image feature points. These feature points identify the robot end effector

¹Fixed-in-workspace cameras which observe both the object and the end effector of the robot.

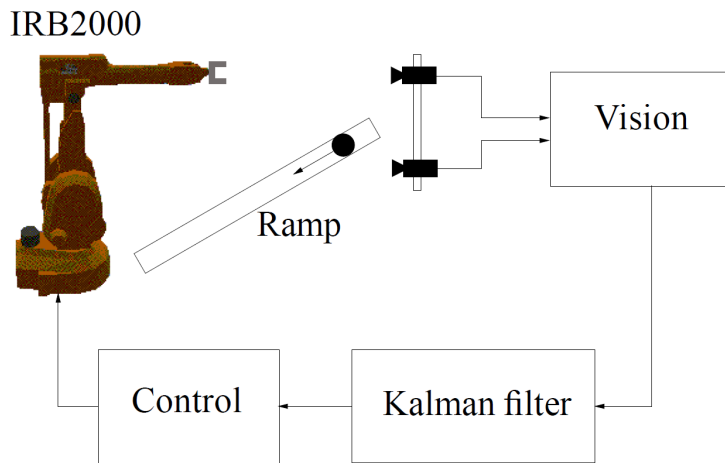


Figure 2.2 The experimental setup consists of a stereo rig looking down a ramp with an Irb2000 robot as end effector forming a visual feedback loop.

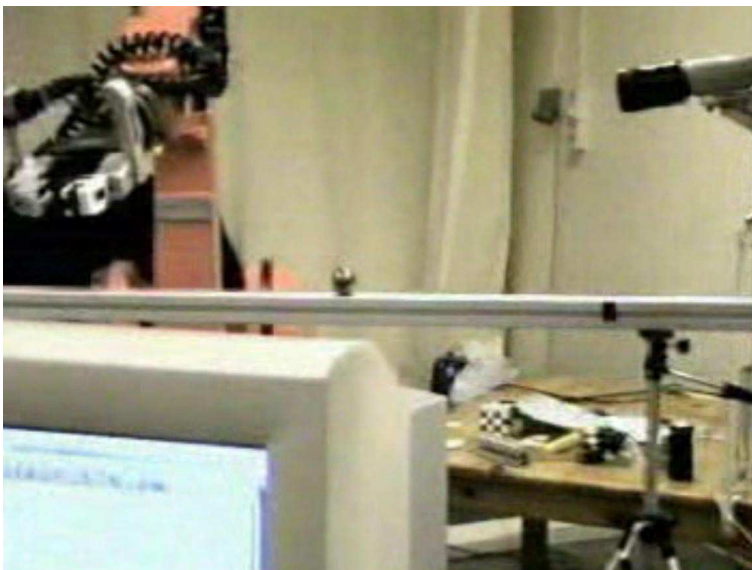


Figure 2.3 The experimental setup. To the right is the stereo rig looking at the robot hand to the left. The object has just been released and is speeding towards the hand. Object speed at gripping point will be approximately 350 mm/s.

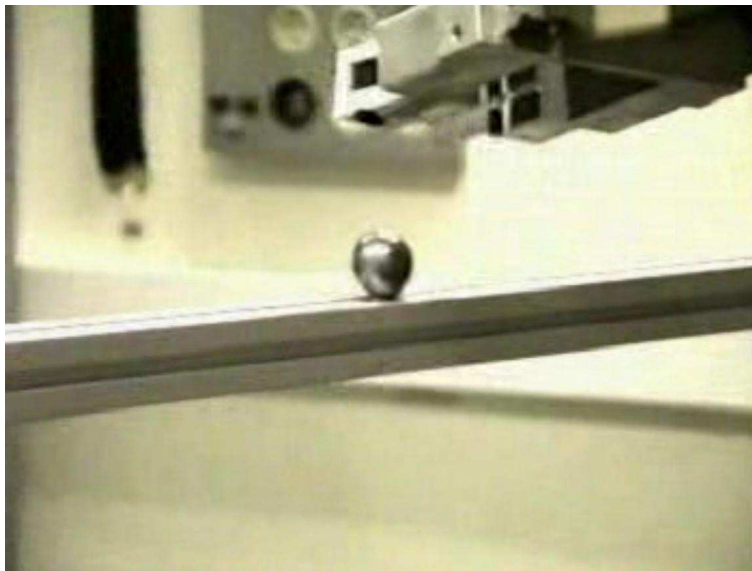


Figure 2.4 Robot gripper is pursuing object and tries to stay a bit ahead to avoid occlusion.

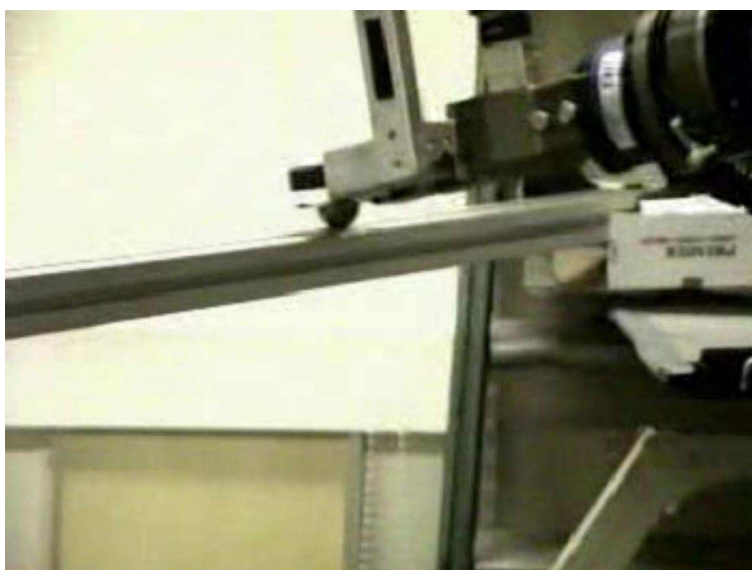


Figure 2.5 Moment of truth. The object is occluded by the robot hand and the grip is performed using predicted data.

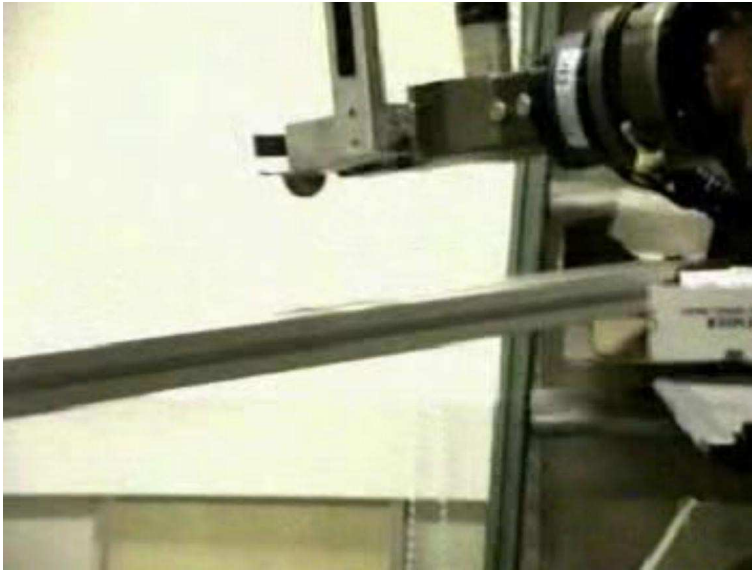


Figure 2.6 The robot hand has successfully caught the object and is performing a pre-programmed break-away movement.

and the target object, in this experiment a rolling ball. The following operations are used to find the feature points in each image [3, 4]:

- Pixel-wise color thresholding into binary image followed by 4-connected segmentation. This is used to locate a marker positioned on the robot end effector. Tracking is performed by locating the closest region having approximately the same size in consecutive images.
- Pixel-wise image difference combined with color thresholding and followed by 4-connected segmentation. This is used to locate the ball moving along the ramp. Tracking of the ball is performed by locating a region-of-interest marked by region size and position.

The difference in corresponding feature point displacements between the robot end effector and the ball are used to control in the stereo rig depth direction.

The camera matrices used for the calculation of the estimated real image ball trajectory were retrieved using methods presented by [5].

Control If the prediction error is small it is possible to apply the separation principle [6], which allows the design of the controller and the predictor to be separated. Assuming the separation principle to be applicable, we design a PI-controller without time-delay to implement the

IBVS control approach.

The inputs to the controller are the predicted robot end effector and ball feature points obtained from the vision system. Each feature point is expressed as 2D coordinates in the image plane of each camera on the stereo rig.

The PI-controller is set to minimize the differences between the robot and the ball feature points for movements in the image plane and the difference between displacement between corresponding feature points for movement in depth.

The output from the controller is a series of move commands, defining trajectories consisting of joint values and joint velocities. We have used this since most industrial robots have interfaces for accepting trajectory move commands.

Time stamping Time stamping is performed on each image from the cameras when they are received on the PC over the FireWire network. This introduces an error of approximately 6ms, $\tau_k^d + \tau_k^{cv}$ (Fig. 2.1). We consider the error to be systematic since the FireWire network is dedicated to each single camera and the network utilization is 18%. It can be compensated for by adding 6ms to each time stamp.

2.4 Results

The system integration and control proved to be sufficient to solve the pursuit-and-capture task in a satisfactory way.

Fig. 2.7 shows the $\tau_k^v + \tau_k^{vc}$ delays for Cameras 1 and 2 in the visual servoing loop. We notice that the delays follow a two point distribution, shifting between 14ms and 28ms for Camera 1 and between 16ms and 34ms for Camera 2. See also Fig. 2.8 for a histogram over the delays.

Fig. 2.9 shows the prediction error for the x image coordinates in Camera 2 and the distribution of the prediction error. The error variance is 0.68 pixels.

Images from the experiment are shown in Figs. 2.3, 2.4, 2.5 and 2.6. In Fig. 2.10 the real, predicted, and measured image coordinates for Camera 2 x-axis are shown. The influence of visual disturbances in the measured data have been almost removed. The predicted data closely follow the estimated real trajectory. In Fig. 2.11 at the time between 21.5-22s, we temporarily lose track of the ball and therefore stop updating the Kalman filter and run the model in open loop. This method works successfully and we are able to follow the ball.

The ball velocity at the gripping point is estimated to 350mm/s and the average velocity to 200mm/s.

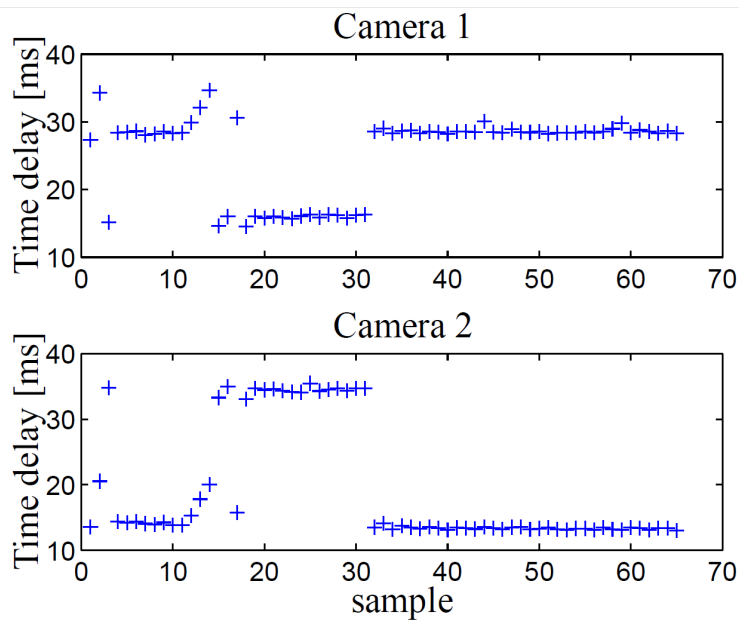


Figure 2.7 Computational and network delay for Cameras 1 and 2.

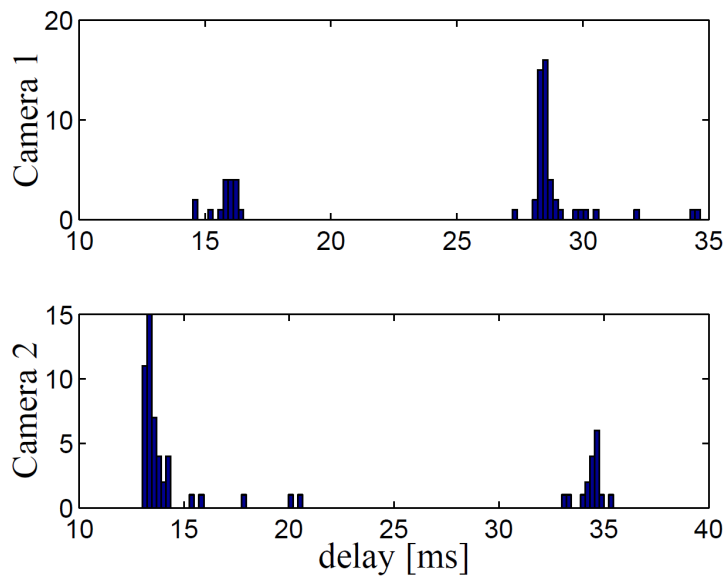


Figure 2.8 Histogram distribution of computational and network delay for Cameras 1 and 2.

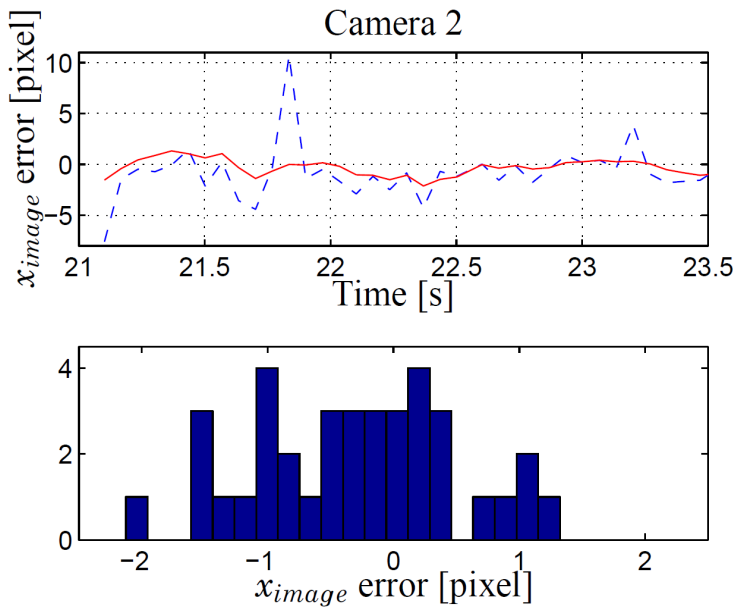


Figure 2.9 Upper figure: Prediction error (solid). Measurement error (dashed), both errors shown for x-axis of Camera 2. Lower figure: Histogram distribution of prediction error for x-axis of Camera 2.

2.5 Discussion

As to distributed control over communication networks, several interesting theoretical problems arise. Time delays may give rise to decreased control robustness, lack of performance and even instability. An advantage of time stamping with prediction is that the separation principle applies [6] – i.e., the design of the controller and the predictor is separated. This allows the use of controller design principles that apply to systems with no time delays as long as the prediction error distribution is taken into account. One approach is to measure this error off-line (or on-line with some delay, forming an outer adaptive loop), and then use this to estimate a noise model, which in turn can be used in the estimation. Our Kalman filter design problem may be viewed as a special case of a very fundamental problem in distributed sensing, computation and control, i.e., the sensor fusion problem. In our case, we deal with feedback data incompatible in time delays.

In the experiment, an ordinary PI-controller based on the predicted error was used successfully. It proved to be robust enough to compensate

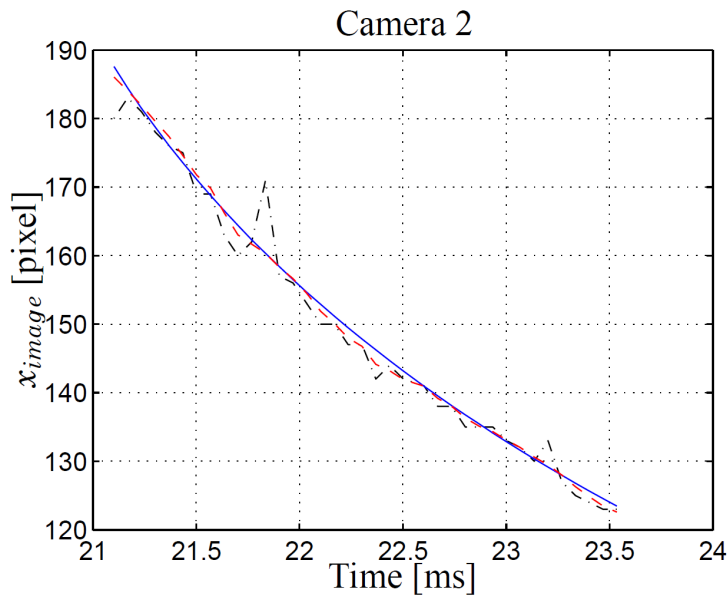


Figure 2.10 Image plane ball trajectory. Estimated trajectory (solid). Predicted trajectory (dashed). Measured trajectory (dash-dotted). Shown for x-axis of Camera 2.

for our prediction error without taking the prediction error distribution into explicit account in the controller design process.

An error was introduced because of the thread scheduling policy of the win32 environment in the PC, where each thread executes in time slots of approximately 20ms. By careful use of the native sleep function, we have managed to suppress the worst-case time-stamp error to approximately 5ms. The worst case will occur rarely since the time stamping threads run with the highest possible priority.

The asynchronously received camera images in the win32 system are one main origin of the two point time delay distribution. The points that deviate from the two point distribution appear to be affected by threads running in the system, for example, the GUI thread and the communication thread. It can be seen that both cameras are affected equally.

Future work

In this paper, we have worked with a fixed Kalman filter. It would be relevant to exploit adaptive Kalman filters. It would also be interesting to try out different noise models.

Phenomena of illumination such as reflections, shadows, time of day

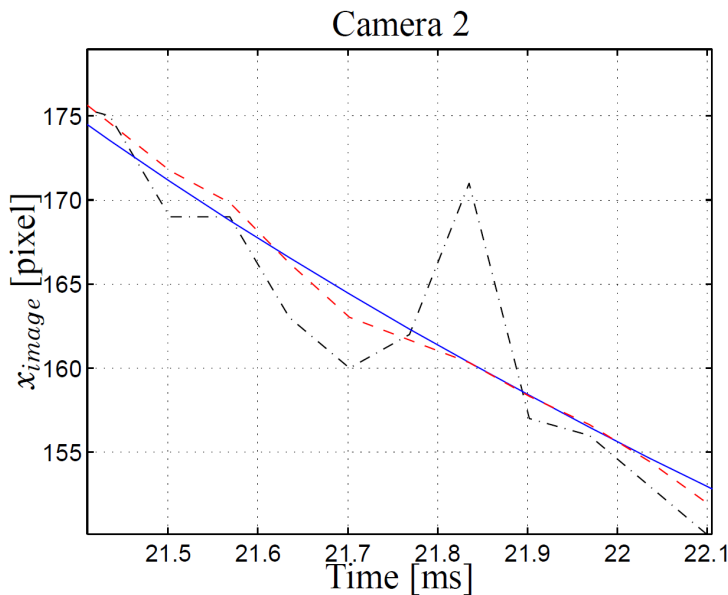


Figure 2.11 Image plane ball trajectory. Estimated trajectory (solid). Predicted trajectory (dashed). Measured trajectory (dash-dotted). Shown for x-axis of Camera 2.

and room lamps affected the visual servoing loop noticeably. Further attempts to increase robustness of the loop properties should take lighting changes into consideration.

Currently, the experiment uses visual servoing feedback on linear movement only. It would be interesting to test the loop against other movements and a wider object speed range, such as a thrown ball.

If increasing the sample rate in the loop even further, the one-step Kalman predictor will not be valid because the time delays introduced will possibly be more than one sample. Therefore, it could be interesting to try out various n-step prediction methods.

More room for improvement might be offered by hidden Markov models and Kalman filters based on more sophisticated models or other observers exploiting properties of finite-state machines and dynamics.

2.6 Conclusions

We have presented how time stamping with nonuniform Kalman prediction compensates for varying time delay errors and image noise errors.

As a result, overall system performance has been found to improve. The prediction error variation was found to be 0.68 pixels. The identified time delay distribution was found to be a two point distribution.

We have experimentally verified this scheme on a visual servoing system running partly on a soft real-time operating system and using non-dedicated network communication to control a robotic gripping operation of a moving object.

3

Extending an Industrial Robot Controller

Enable networked sensor-based motions

3.1 Introduction

Many promising robotics research results were obtained during the late 1970s and early 1980s. Some examples include Cartesian force control and advanced motion planning. Now, 20 years and many research projects later, many technologies still have not reached industrial usage. An important question to consider is how this situation can be improved for future deployment of necessary technologies.

Today, modern robot control systems used in industry provide highly optimized motion control that works well in a variety of standard applications. To this end, computationally intensive, model-based robot motion control techniques have become standard during the last decade. While the principles employed have been known for many years, deployment in products required affordable computing power, efficient engineering tools, customer needs for productivity/performance, and improved end-user competence in the utilization of performance features.

However, applications that are considered nonstandard today motivate a variety of research efforts and system development to package results in a usable form. Actually, robots are not useful for many manufacturing tasks today, in particular those found in small and medium enterprises (SMEs). Reasons include complex configuration, nonintuitive (for the shop floor) programming, and difficulties instructing robots to deal with variations in their environment. The latter challenge includes both task definitions and definition of motion control utilizing external sensors. The key word here is flexibility, and flexible motion control is particularly

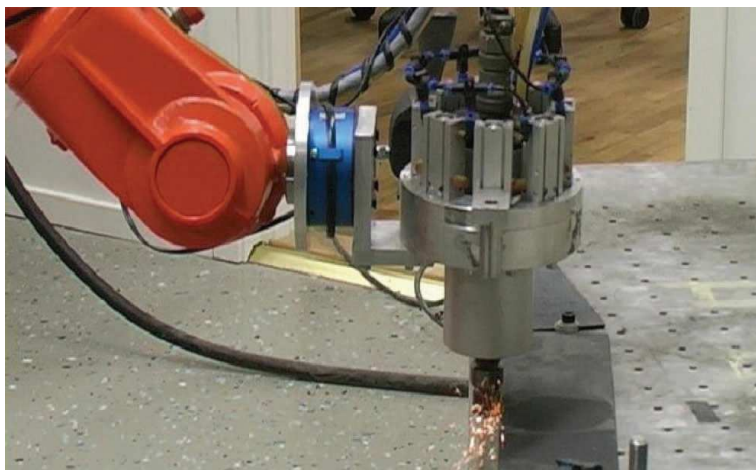


Figure 3.1 High-bandwidth, force-controlled grinding using an ABB IRB2400 industrial robot with the extended ABB S4CPlus control system.

difficult since the user or system integrator needs to influence the core real-time software functions that are critical for the performance and safe operation of the system. We must find techniques that permit real-time motion controllers to be extended for new, demanding application areas.

Open Control

Most robot control systems today support some type of user inputs/outputs (I/Os) connected on local networks or buses. A crucial issue is the achievable bandwidth for the control loops with the external feedback. For many applications, the effect of the bandwidth limitations only shows up at longer duty cycles, whereas for some applications (like contact force control between the robot and the environment/workpiece; see Figure 3.1), stability problems and severe performance degradation may result [7], [8], [9].

Viewing robotics research from a control perspective, direct access to the driving torques of the manipulator and fast feedback are very valuable, or even crucial, for algorithm evaluation and implementation of high-performance control schemes. This made early robot systems like PUMA 560 popular. Unfortunately, this kind of low-level access is not present in the commercial robot control systems of today. The difference is that today we should not only be able to close fast feedback loops at a low level, we also need to do so in a consistent manner, supporting supervision and coordination with the application-oriented programming on higher hierarchical levels. Therefore, alternative ways to obtain high-bandwidth control

based on external sensors, which maintain the existing supervision and coordination functionality, are necessary.

An examination of five major European robot brands (ABB, Comau, Kuka, Reis, and Stäubli) shows that they all, to some extent, provide support for application-specific motion control. Some controllers are fully open but only if all original safety and programming features are disabled. In the project considered in this article, we have used the ABB S4CPlus controller as an example. Whereas S4CPlus is not an open system, its internal design provides some features for development of open control. Similar results have been reported for other systems (see, for example, [10]).

Open Issues

Developments up to the current state of the art raise fundamental questions that form the motivation of this article.

- Today, industrial robot controllers provide highly optimized, model-based motion control that claims to be fully programmable and configurable. Still, when new autonomous or service robot systems are developed, systems developed for industrial manipulation are hardly ever used. Instead, manipulator control is redeveloped but without the full performance and system robustness that would be possible if results/systems from industrial control were used. Can current industrial controllers be useful as components in future advanced robot systems?
- Twenty years ago, proceeding from a textbook algorithm to a functional implementation required extensive engineering efforts. Today, we have engineering tools and code generation from specifications, descriptions, and simulations of control principles. Comparing experimental work within the academic community with industrial robot development, engineering tools such as MATLAB, Maple, and the like are quite similar, whereas the code generation and deployment of controllers/components appear to be quite different. Deployment in a product requires substantially more verification, optimization, and tailoring to the system at hand. Then, the question is this: Could commercial/ optimized systems be structured to permit flexible extensions, even on a hard, real-time level?

Objectives

We try to answer these questions by confronting theoretical and experimental laboratory results with actual industrial reality. A most challenging case, also representing the 20-year lag between experiment and

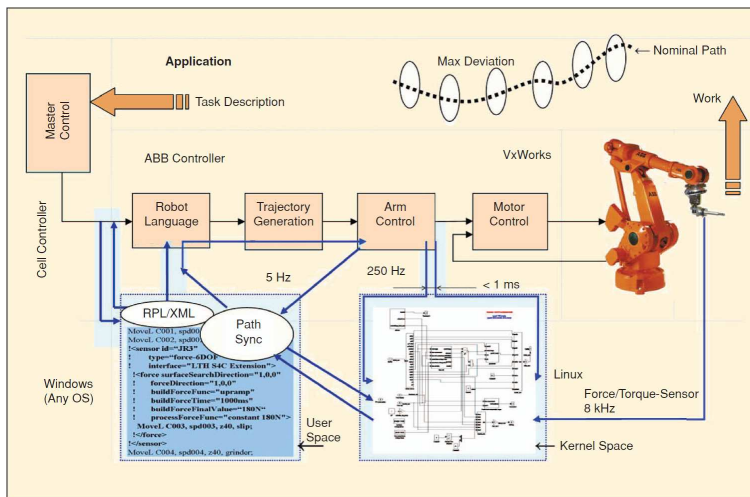


Figure 3.2 The extension of an industrial robot controller with sensor interface and support for external computations and synchronization. The configuration uses a Motorola PPC-G4 PrPMC-800 processor board mounted on a Alpha-Data PMC-to-PCI carrier board with a local PCI bus.

product, is high-performance, 6-degrees-of-freedom (DOF) control of the contact forces between the robot and its environment. As a part of the offline automated fettling and finishing (AUTOFETT) project, where the main objective was to develop flexible support and handling devices for castings, force-controlled grinding was accomplished and brought to industrial tests; this will serve as our primary example.

We will consider different aspects of incorporating a fast “sensor interface” into an industrial robot controller system, where the ABB S4CPlus system will be taken as the primary example. The term sensor interface may be a bit restrictive because, in practice, the interface not only allows for feedback from external sensor data but also allows for code and algorithms to be downloaded and dynamically linked to the robot control system (Figure 3.2).

3.2 Considerations and Design of System Extensions

The architecture of the ABB S4CPlus control system and its extensions are shown in Figure 3.2. Task descriptions, as given by the robot programming language RAPID, are passed through the trajectory generation and turned into references for the low-level servo controllers. Extensions to

the system, based on present and future applications requiring the use of external sensor-based control, could be made by modifying references on any level (task, Cartesian, joint, or motor currents level). We will discuss the underlying design considerations and our implementation of the platform.

On a high level, the present ABB S4CPlus (and earlier) systems already feature the ability to read sensors via customer I/O to influence the robot task as expressed in programs written in the ABB RAPID language. The RAPID program reading sensor information via the I/O system can be referred to as a pull protocol, which requires no external computing. The sensor reading/handling, however, must be expressed in the user program. Today, it is also possible to change programmed motion targets via remote procedure calls (RPC) during robot motion, which can be referred to as a push protocol. This requires external computing but less RAPID programming since the logic of how sensor data should influence motions is expressed in external software. Both these alternatives are of great value and should be maintained, but there are also two major problems that must be resolved in future systems.

- **Performance:** Restricting the use of external sensors to the RAPID level only implies that new types of high performance motions cannot be introduced with a reasonable engineering effort. Some simple cases have been solved, such as the control of external welding equipment, but the fundamental support for motion sensing is missing. Whereas force control is much needed within several application areas, such as foundry and assembly, it is currently quite difficult to accomplish in the robot work cell.
- **Flexibility:** The use of port-based I/O data without self description leads to less flexible application programs that require manual configuration, limiting development of high-level application program packages.

Therefore, today we have high-level (user-level) usage of low-level (primitive) sensors. To overcome the two aforementioned problems, we also need low-level (motion control) usage of high-level (force, vision, etc.) sensors. As a first step, interfacing with force sensors should be supported. This is both a technically demanding case and a desired one from a customer point of view.

With this overall goal, some specific topics will now be covered.

Hardware and I/O Protocol

The most promising hardware interfacing possibilities (from a cost and performance point of view) are shared memory access via the peripheral

connection interface (PCI) system bus and standard high-speed Ethernet communication. To some extent, these techniques are already used in S4CPlus.

Shared Memory Obtaining sensor data directly in shared memory simplifies system development since the system programming model is unchanged. Shared memory is assumed to be provided via the PCI bus. S4CPlus already supports installation of PCI plug-in boards, although this feature is not made available to customers.

Presently, most sensors do not come with a PCI interface, even if some simple sensors can be connected to PCI-based I/O boards. However, some advanced sensors, such as the JR3 force-torque sensor, provide a PCI-based interface. The trend is that more and more PCI-based sensors are becoming available. This method of interfacing external sensors also allows for adding “intelligent sensors” (sensor fusion or sensors with additional computational power).

Networked Sensor interfaces can also be networked based on field buses, which are available on the user level for all modern controllers and on the servo level for some controllers. However, it appears that field-bus interfaces and communication introduce delays and limited performance compared to the shared memory interface.

As an alternative, our experiences with Ethernet communication using raw scheduled Ethernet or UDP/IP show promising results [11]. The bandwidth is comparable to that of the PCI bus, and the standard network-order of data bytes simplifies interfacing. Also, with proper network/interrupt handling, the latency can be very short, showing great potential for future applications utilizing distributed sensors.

Safety and Quality Issues

Open systems require careful engineering to avoid exhibiting unpredictable or even unsafe behavior when confronted with inexperienced users and extended with novel features at the customer site. One significant challenge in the development of open systems is the complexity in the systems engineering, where several difficulties, which are discussed below, must be addressed.

Hardware Reliability Installing third-party hardware means there is an additional risk for system failures, despite the high and ensured quality of the basic robot system.

The added hardware may fail without affecting the robot hardware, but it can still lead to system failure from an application point of view if

the application was made dependent on the added hardware. Also, third-party modules may severely interfere with the communication on the data buses used by the control computers. Such a failure can be due to faulty added hardware, to bad configuration, or to incorrect access of the bus interface of the added hardware.

To avoid these problems, customers or in-house application developers should write the application software in such a way that functionality can be tested based on some dummy sensor data without using the actual hardware. This can also be accomplished by running the application with a virtual controller. Hence, guides for developing sensor-based applications could and should be supported within graphical robot simulation and programming tools.

Sensor failures are inevitable and have long since been an important obstacle in real applications. In cost-efficient production, it is not as simple as saying that there should be redundant sensors; this configuration is costly and increases the risk of system overload/failure. A combination of system structure, proper interface design, testing methodology (including simulation support), and well-defined fall-back control is needed.

Data Integrity A serious problem, from a safety point of view, is the risk of external software damaging important robot system control data (for instance, due to bad pointers or bad array indices in the external software). Therefore, common data areas should normally be located on the added board and then accessed by the robot controller.

Classified System Properties The definition of shared data (control signals and other internal states) could be achieved by providing available header files. However, using the ordinary header files would expose potentially classified motion control techniques in too much detail. Therefore, there should be a neutral definition of (possibly) exposed variables, preferably based on information from textbooks or articles and possibly suggested as an open standard.

Robot Safety Even with hardware and software functioning as intended (as described in previous sections), in a strict technical sense, there is a potential risk that the external logic interacts with the control logic in an unforeseen manner. That is, even if the externally added software does what the program states, it can potentially still compromise robot safety functions.

To overcome this difficulty, the states exposed to external software should be copies of the internal true state, and external states need to be cross-checked before influencing the modes of the standard robot control. Updating can be periodic in some cases, whereas other states (such as

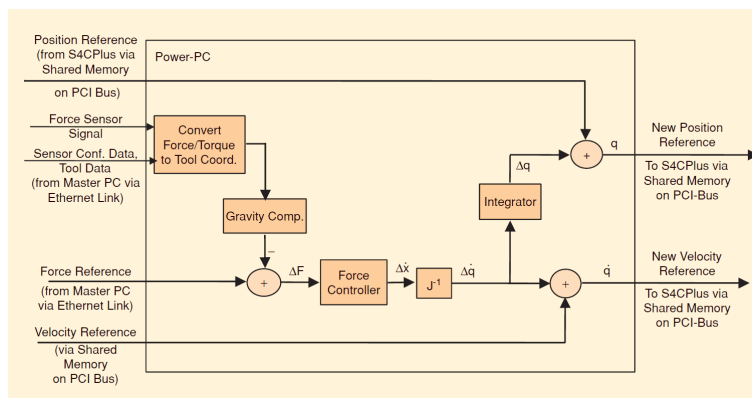


Figure 3.3 The force controller block structure. The force control algorithm is implemented inside the block labeled *Force Controller*.

run-mode and brake states) should be updated in an event-driven fashion in order to improve consistency between internal and external states, including generation of interrupts to the external software.

Perhaps the most important part of safety is the ability to keep the internal safety functions activated (possibly with adjusted tolerances), even during sensor-based motions. While this problem has been solved, the remaining challenge is to combine safety with performance.

Performance

For industrial robots, control performance means productivity. Specific force control algorithms (inside the Force Controller block in Figure 3.3) are outside the scope of this article, but the imposed requirements on the open system deserve some attention.

Sampling and Bandwidth Considerations As an example, force control in a noncompliant environment typically requires fast sampling since excessive contact forces may build up very quickly, for instance, during the impact phase. It is also well known from control theory that feedback from a sampled signal decreases the stability margin, thereby decreasing the robustness to varying operating conditions.

In the architecture of the ABB S4CPlus system, there are a number of levels in which external control actions can enter the system. First, high-level feedback using the high-level ABB RAPID language to modify the generated trajectories gives a sampling time of $h = 0.1s$. The interface to the builtin arm servo control has a higher sampling frequency, $h = 4ms$. Finally, $h = 0.125ms$ gives the maximum internal sampling frequency of

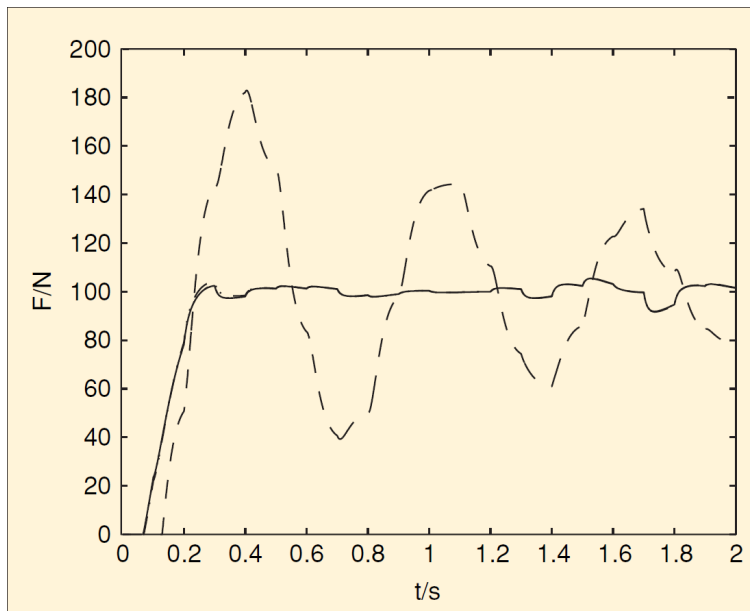


Figure 3.4 The contact force for continuous time design (solid), $h = 0.1s$ (dashed), $h = 4ms$ (dash-dotted), and $h = 0.125ms$ (dotted).

the JR3 force/torque sensor.

A simple simulation will illustrate the effects of different sampling intervals for a highly simplified model of a typical force control task. A linear model with 1 DOF of a controlled robot is given by the transfer function model $F(s) = 35000k/(20s^2 + 1500s + 35000)X_r(s)$, where F is the contact force, k is the stiffness, and X_r is the commanded position reference.

In Figure 3.4, we can see the results of the simulation using a rough surface of stiffness $k = 25N/mm$, with a continuous time proportional controller as well as discrete time control with the sampling times described above. The desired force F_r was 100N. It can be seen that the response for $h = 4ms$ is almost identical to the continuous time design, thanks to the fast position control in the inner control loop. The 4-ms level has been determined to be a good trade-off in many force control applications, considering also the limited available computational power. However, in some applications, such as force control in extremely stiff environments or applications where high approach velocities are required, a sampling rate higher than 4 ms may be desired.

User and System Aspects

Open controllers need ways of expressing the usage of sensors. The RAPID language of the S4CPlus controller supports sensor feedback through a RAPID concept called correction generator. This language mechanism allows the robot program to correct the robot path during operation with information typically derived from a sensor.

Unfortunately, the built-in RAPID mechanism is not applicable for use by force sensor feedback, primarily for two reasons. The correction generators only support position-based path correction, while force feedback may require torque-based path correction. Second, the update bandwidth supported is much too low to apply to most force control applications. Whereas some servo-level extension is needed to accommodate the bandwidth requirements, the user programming level requires extensions for force control.

Language Extensions The specification of desired force control should be available on the user level, where the rest of the robot application is specified. The solution was to introduce two new language scopes into the RAPID language, integrating handling of sensor-influenced trajectories into the language itself. In order to be backwards compatible with standard RAPID, the new code was encoded as XML scopes and tags within RAPID comments (Figure 3.5). The processing of the XML comments is conducted in a new master PC module acting as a robot proxy (see Master PC in Figure 3.2), which then communicates both with the original program server of the S4CPlus controller and with the added low-level control on the added PCI board.

System Connections The communication between the master PC and the S4CPlus controller is over the Ethernet using TCP/IP and UDP/IP. This is not within the force control loop as such; its purpose is to synchronize the robot program execution with the lowlevel force control along the programmed path. This was accomplished by the following design elements:

- The force scopes in the ExtRAPID program are replaced by calls to a generic motion-server, which is written in RAPID and downloaded with the rest of the application. The force-controlled MoveL instructions are kept in the master PC (see Figure 3.2) and fed to the program server of the S4C controller via the ABB Robot Application Protocol (RAP).
- The embedded motion server carries out the motions by executing TriggL instructions (instead of the original MoveL) with extra arguments that form a subscription of an I/O byte output. Later, when

```
MoveL C001, spd001, z40, grinder;
MoveL C002, spd002, z40, grinder;
!<sensor id="optidrive"
!     type="force"
!     interface="LTH+ABB S4C Extension">
!   <force surfaceSearchDirection="1,0,0"
!     forceDirection="1,0,0"
!     buildForceFunc="upramp"
!     buildForceTime="1000ms"
!     buildForceFinalValue="150N"
!     processForceFunc="constant 150N">
      MoveL C003, spd003, z40, grinder;
!   </force>
!</sensor>
MoveL C004, spd004, z40, grinder;
```

Figure 3.5 A sample ExtRAPID program. The extended language constructs are located in RAPID comments and are modeled as XML tags in order to be easily modifiable.

the S4C servo actually performs the motion, that output (the sync signal from servo to master PC in Figure 3.2) forms the lower byte of the integer value of a system-wide path coordinate.

- The master PC uses the received path coordinate as the basis for the 4-ms advancement along the path, maintaining the overall path coordinate and computing force control set points and parameters accordingly.

Due to the limitations on buffering according to the first design element, and since an external set point in real time can influence the set points to the force control loop (Figure 3.2), any external sensors connected to or communicating with the master PC in real time can be used for instant feedback to the motion control. Note that the ABB controller is then kept aware on the top level of the robot's commanded target.

External Motion Control With language extensions and system connections in place, the implementation of the actual external controller (to the S4C) can be accomplished. This is the force control in Figure 3.2. To accomplish hard, interrupt-driven, real-time execution with shared memory communication, the force controller is run as a Linux kernel module. Such a module can be replaced without rebooting the system, but programming for kernel mode is a complication. However, all parts of the force controller (including the shared memory interfaces) were implemented in C as Simulink blocks, which (apart from being used for simulating the system) were cross-compiled to the target computer and incrementally linked to form a Linux kernel module. The porting of the Linux kernel to the specific computing and I/O hardware was carried out in our laboratory as was the tailoring of the build procedure for making Linux kernel modules for sensor feedback.

The host computer version of the Simulink blocks are first translated for embedding by using MathWorks Real-Time Workshop, then compiled and linked with external libraries. In the resulting system, the control engineer can graphically edit the force control block diagram and then build and deploy it in the robot controller.

Simulation

Apart from simulating the force control as such, it is also highly desirable to be able to simulate sensor-based robot control from an application point of view.

Simulation of Low-Level Control Designing the controllers using MATLAB/Simulink (which as described above also gives the implementation) means that the Simulink models can also be connected directly to existing models of the robot; furthermore, models of the environment and sensors can be used for simulation. Tools such as Modelica and Dymola were used and proved to be very effective for the modeling and simulation of many types of dynamical systems, including industrial robots [12], [13], [14], [15].

External Sensing in the Digital Factory Traditional off-line programming does not use the full potential of virtual models and simulation systems in industrial robot applications. The interface between the off-line programming system and the robot controller is today restricted to program transfer. Considerable improvements have been made in the accuracy of programs created off-line, especially since the introduction of technologies such as RRS (Realistic Robot Simulation; <http://www.realistic-robot-simulation.org>). However, extensive problems remain. For instance, when high-level sensors such as vision, force, and laser scanning are used,

no mechanism is available to relate the sensor information to prior knowledge actually existing in the model created in the off-line system.

If the virtual model could be accessed during the execution of the robot task, intelligent decisions could be made despite changes in the state of the robot work cell that were not anticipated when the robot task was planned. Instead of using a simple feedback loop to the robot movement, the virtual model is continuously updated, allowing new information and previous knowledge to be accumulated in a common format. High-level replanning of the robot task can then be automatically performed. Typical limitations of robot systems that are hard to handle online include collisions due to obstacles unknown to the robot program and deviations of the setup and kinematic singularities during linear movements. Successful implementation and experiments have been made in the present case project.

3.3 High Power Stub Grinding

Applications such as stub grinding and deburring represent good examples of common tasks which would be desired to automate. Not only are manual fettling and finishing major cost elements in the production process (representing up to 40% of total costs) which often lead to inconsistency in quality and delivery delays, there are also severe health aspects related to this process in today's foundry industry. In these types of material removal applications there are two important issues, accurate control of the force between the tool and the work object, and starting and ending the material removal process without lowering the quality of the machining. In order for an industrial robot to be able to satisfy these requirements, functionality for high-bandwidth contact force control needs to be implemented in current industrial robot systems.

Force control structures and application requirements

During the tasks considered, only the motion perpendicular to the surface of the workpiece was required to be force controlled, and therefore a hybrid force/position control strategy [16] was employed. In this type of structure one or several degrees of freedom become force controlled, while ordinary position control is used in the other directions. The force-controlled direction is perpendicular to the surface as required, while the motion tangent to the surface and the orientation are typically controlled using position control. The directions which should be force controlled are selected using a diagonal selection matrix, which is set as a part of the high-level task specification. The block structure for the controller is



Figure 3.6 Force controlled stub grinding using an ABB Irb6400 industrial robot with an extended ABB S4CPlus control system.

shown in Fig. 3.7. The controller creates Cartesian trajectory corrections by filtering the contact force through a MIMO linear transfer function matrix $G(s)^{-1}G_I(s)$, where

$$G_I(s) = (Ms^2 + Ds + K)^{-1} \quad (3.1)$$

is a second order impedance relation and $G(s)$ is a decoupled linear MIMO first-order approximation of the robot dynamics from position references to position in closed loop, chosen to compensate for the robot dynamics by decreasing the phase lag. The parameters in the impedance relation $G_I(s)$ will depend on the active directions in the selection matrix, with infinite stiffness in non-active directions.

Development of an end effector for stub grinding

Software tools and process models need to be developed for the stub grinding application, which could also be used in other applications such as deburring and polishing.

The development of a hydraulically driven end-effector tool for the stub-grinding process was carried out with the understanding that the same tool will be used for surface finishing operations. A “cup” type of

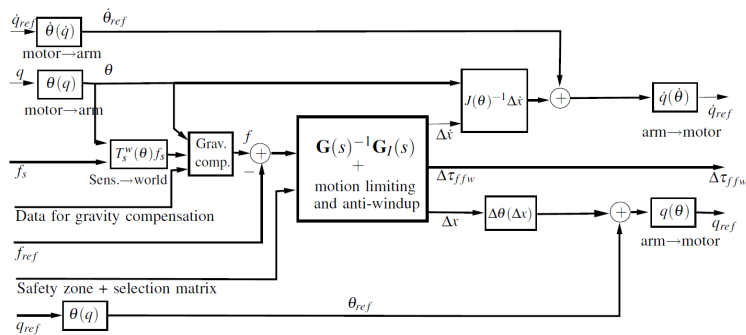


Figure 3.7 Block structure a general hybrid force/position- or impedance controller. The controller calculates a Cartesian reference correction Δx by filtering the contact force through the MIMO linear transfer function matrix $G(s)^{-1}G_I(s)$, where $G_I(s) = (Ms^2 + Ds + K)^{-1}$ and $G(s)$ is a linear MIMO first-order approximation of the closed-loop robot dynamics. A dynamic feed-forward torque signal $\Delta\tau_{ffw}$ can be computed based on a non-linear dynamic model of the robot. The corresponding control code is generated from Simulink/Real-Time workshop and downloaded to a co-processor added to the robot controller.

disc was selected for the stub grinding fettling operation, which reduced the effect of wheel wear to essentially one direction and allowed a greater wheel force to be applied during metal removal.

The main components of the end-effector are the hydraulic motor including the grinding stone, and the force sensor consisting of a displacement sensor and a spring (Fig. 3.8). The stiffness of the springs was set to 60 N/mm but could be increased by a simple replacement of the spring, the stroke being 20 mm. The system's own stiffness was sufficient to react on irregularities of the casting product and surface roughness, which pass at high frequency. Bigger burrs and changes of the contour at lower frequency were handled by the force control system.

Off-line programming

For the machining of complex parts (e.g., stub-grinding of castings) in low series, traditional teach-in programming is not feasible. As done for classical multi-axis milling machines, CAD/CAM programming has been proposed for off-line programming. Here, a more advanced scheme was worked out as shown in Fig. 3.9. Based on the CAD-model, the CAM system generates the tool trajectory, described by a series of tool postures expressed in a work piece coordinate system. Although CAM systems with specific grinding functionality are not commercially available, similarities with classical profiling and surface contouring operations (milling) exist. Therefore, the different tool paths for stub-grinding and finishing oper-

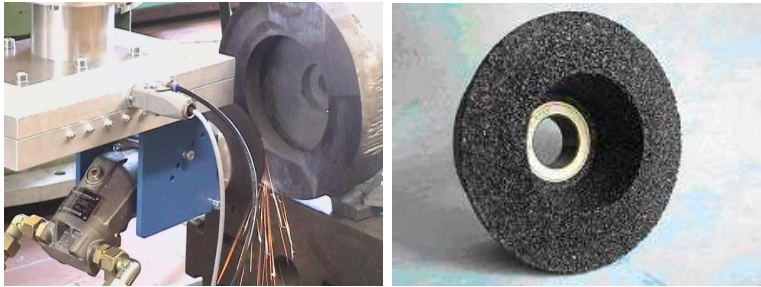


Figure 3.8 The developed compliant grinding tool (left) and cup stone (right) for stub-grinding application.

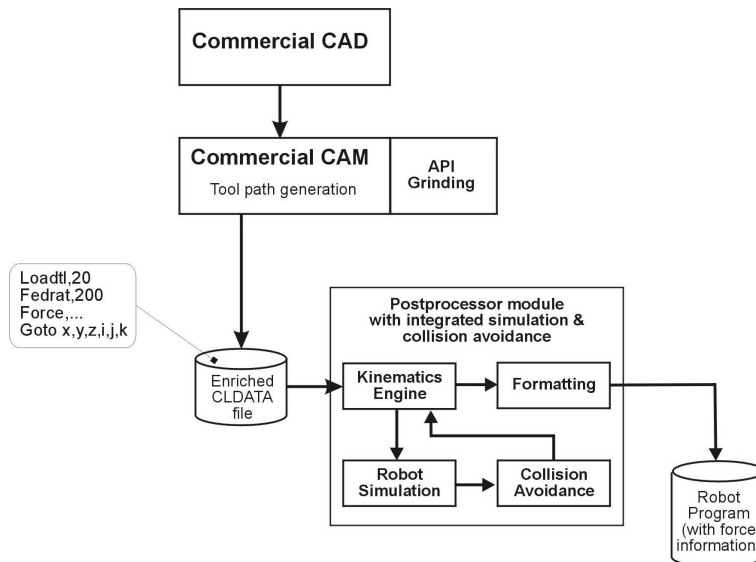


Figure 3.9 Structure for off-line programming, based on CAD/CAM.

ations were initially generated using existing tool path generation algorithms for milling. An "add-on" feature was developed and implemented to generate specific information related to the force controlled robot operation.

Within this research work, an advanced post-processor with integrated robot simulation and collision avoidance functionality was developed, based on an existing off-line robot programming system developed by KPS/Rinas. This concept has been developed earlier, see for example [17]. Each post-processed position was directly checked for collision and if it occurs, a

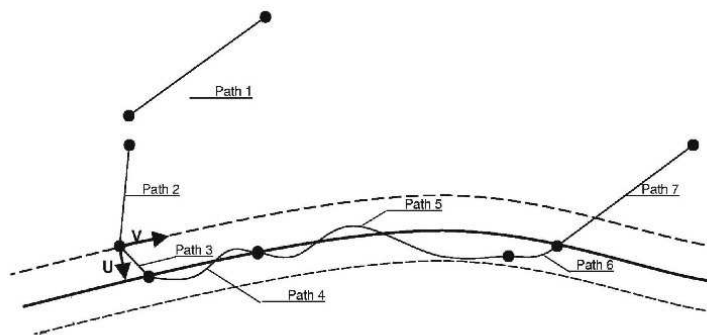


Figure 3.10 Grinding path with combination of force and position control.

collision avoidance algorithm based on the redundancy of the robot system was applied. This means that the post-processor looks for another collision free robot configuration giving the same coordinates.

Grinding strategy

A robust strategy for force control was developed and implemented, using a combined strategy of force and position control. A strategy based on force control can only give problems during entry and exit of the operation. Therefore, the programmed path can contain different sections. Each section can be characterised by different values for speed, change of speed, force and change of force can be defined on different sections of the programmed path. Entry and exit are examples of such sections, and make a smooth transition of non-cutting to cutting movements possible. More specifically, the grinding task is broken down into five phases called approach, force build-up, process, decline and withdraw (Fig. 3.10).

Applications and Experiments

Final experiments were done in a specially developed fettling cell. The cell consisted of the robot (with end-effector), a flexible clamping unit and a control system. Images from stub-grinding experiments and finishing grinding experiments on a propeller blade are shown in Fig. 3.11. The resulting contact force from a stub grinding experiment with reference $F_r = 160N$ is shown in Fig. 3.12.



Figure 3.11 Stub-grinding (left) and finishing on a propeller blade (right).

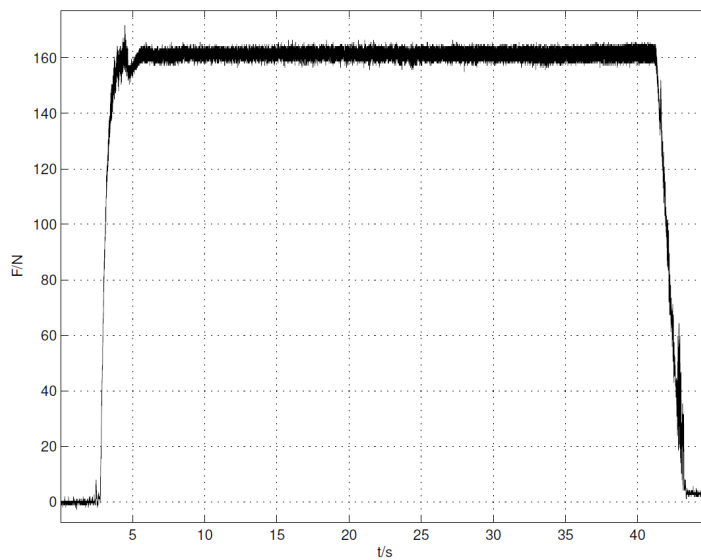


Figure 3.12 Contact force during grinding experiment with reference 160 N.

3.4 Discussion

The fact that robots today handle fully structured and specified tasks in industry very well, and the lack of experience/ knowledge from small-scale manufacturing within the research community, have created the misconception that “industrial robotics is solved”. In future manufacturing, however, the increased need for industrial robots that (typically in

small enterprises) understand human instructions and are able to handle larger task/work-piece variations is apparent.

Force control can be carried out on two levels:

- Forces are controlled by a separate external tool: This means that the adaptation of the force occurs without intervention of the robot. The robot executes a preprogrammed path, and the changes in the tool position to achieve a constant force, are carried out by the end-effector integrated with some additional external axis.
- Forces are controlled by using the robot, which was our chosen solution described in this paper.

The advantages of the robot with integrated force control include the possibility of obtaining higher stiffness at a considerably lower weight, as well as increased flexibility in mounting and accessibility due to the six degrees of freedom.

The close cooperation and technology transfer between industry and academia have been instrumental during the development of the platform, since control and software need to be tightly integrated for performance and applicability. Robotics is multidisciplinary and researchers from many fields and different university departments have been active in the development of the field.

The accomplished sensor interface, we believe, is unique due to the combination of

1. A shared memory interface to the built-in motion control, enabling fast interaction with external sensors;
2. Integration of high-level and low-level control in such a way that low-level instant compensation (within the tolerances of system supervision limits) propagates to higher levels of execution and control, providing state and path coordinate consistency;
3. The external sensing and control is built on top of a standard industrial controller with (due to the previous item) the built-in system and safety supervision enabled, making it possible for the end-user to use all the features (language, IO, etc.) of the original system;
4. The add-ons to the original controller can be engineered (designed and deployed) by using standard and state-of-the-art engineering tools, thereby bridging the gap between research and industrial deployment of new algorithms;

A relevant issue would be to compare different systems and what feedback control performance they provide in a programmable and applicable

manner. To our knowledge, there are no other systems that provide the same high sampling rate and low input-output latency (within a factor of ten) together with all the user-programming features and supervisory functions. Therefore, comparisons with other systems are not meaningful within the scope of this paper, and the focus here is on the application needs and how they were satisfied. Experiences from the fully developed prototype and its industrial usage confirm the appropriateness of the design choices, thereby also confirming the fact that control and software need to be tightly integrated.

The new sensor platform may be used for prototyping and development of a wide variety of new applications. It also offers an open experimental platform for robotics research explored on many hierarchical levels (from control algorithms with high bandwidth to robot programming and task modeling with on-line sensor information). The preserved high-level support and the integration with the supervision and safety system of the standard industrial robot system constitute a major difference to most "Open Robot systems" previously reported, although many robotics labs have reported activity on open control systems which satisfy the need for the above mentioned aspect on evaluation and implementation [18], [19].

3.5 Conclusions

This paper describes the design and implementation of a platform for fast external sensor integration into an industrial robot control system (ABB S4CPlus). An easily reconfigurable control structure was achieved, which is able to control contact forces with a sampling bandwidth of an order of magnitude higher than for conventional robot control systems. As motivating examples of industrial applications, the implementation of force-controlled grinding and deburring are described. Additionally, a new end-effector was developed and integrated with the robot control system.

Furthermore, experiences from this project has lead to the industrialization of force control for assembly, grinding and deburring applications at ABB.

4

Case Study: Cost-Efficient Drilling

Improved flexibility by substantially improved accuracy for high value products

4.1 Introduction

The traditional application areas for industrial robots involve highly repetitive operations such as spot welding. Hence, robotic development has been focused on high precision (repeatability) in repetitive operations. For example, a standard ABB IRB4400 robot for 60 kg payload has a repetitive accuracy of $\pm 0.05mm$. If, however, the same robot is given a new coordinate that it has never visited before, the accuracy is in general around $\pm 3mm$, which is 60 times the size of the repetitive accuracy. Today it is possible to buy the same robot with an option pack that includes calibration for high accuracy. This will improve accuracy to become within $\pm 0.5mm$, which is still 10 times the repetitive accuracy. In addition, this accuracy is only guaranteed under the condition that no unmodeled external forces act on the robot, i.e., only during motion in free space. For contact tasks such as polishing, drilling, and riveting, the effects of the limited mechanical stiffness of the robot must be taken into account in order to maintain the positioning accuracy, which is not feasible unless a detailed model of the particular robot specimen is available.

Systems for automatic drilling have a long history both in industry and the research community. In particular, the use of industrial robots for drilling is interesting due to their flexible programming and the comparatively low cost of industrial robot systems. However, robot drilling is a very challenging task due to the comparatively low mechanical stiffness of the typical serial industrial robots in use today. In general, clamp-up is

necessary in robotic drilling to avoid vibration during the drilling process as the drill tool generates vertical, horizontal and axial forces during the cutting process. The compliance makes the robot deflect, sometimes up to several millimeters, due to the externally applied forces during clamp-up and drilling. Due to the bending of the robot links and the elasticity in the gears, the local deflection at the contact point does not necessarily occur in the (axial) direction of the applied force, but may have tangential components which are on the same order of magnitude as the axial deflection. This tangential deformation results in poor hole quality and inaccurate positioning. In contrast, aerospace tolerances require drilled holes to be accurate within $\pm 0.2mm$ [20]. A drilling process involves moving a drilling end-effector to the correct position of the hole. Prior to drilling, a pressure foot is used to press the parts together in order to avoid burrs entering in between the plates. In addition, the pressure foot assures that the drilling machine is kept stable throughout the drilling cycle. A self-feeding mechanism is normally used to feed the drill through the stack of materials. Automated drilling in the aerospace industry today uses large robots for two major purposes: to handle the large assemblies, and to accurately counter balance the drilling forces involved in the drilling process. There are many different ways to overcome forces in drilling and fastening using industrial robots. One approach is to divide the process in two steps, where in the first step the robot stiffness is mapped by applying forces to the robot TCP and measuring its deflection, while in the second step the robot is adjusted back to the nominal position under load. These compensation values are then applied as a filter to program the robot during process execution [21]. Mapping the robot stiffness in this way can, however, be extremely time consuming. Other methods to solve the skating problem have been tested by using metrology systems to supervise the robot, see for example [20], [22], [23], [24]. In such methods, a metrology system is connected to the robot controller via an external feedback loop to update the nominal position of the robot with measurements in real time. However, using metrology is not a straightforward solution, as the robot will deflect as the pressure foot is engaged.

Common to the traditional approaches is the lack of high performance sensor feedback to the robot, whereby the robot cannot be updated fast enough to cope with the dynamic process that drilling involve. Robot control systems are traditionally closed, a circumstance which has hampered system integration of manipulators, sensors and other equipment, and such system integration has often been made at an unsuitably high hierarchical level. As a more cost-effective solution, high bandwidth feedback techniques can be used to control the properties of the drilling process. Research and development on force-controlled drilling has not received as much attention as many other applications of industrial force control,

such as assembly, deburring, milling or polishing. The reason is probably the difficulties involved in robotic drilling, and the lack of available industrial robot systems with capacity for high-bandwidth force control. Some results on force control for special drilling machines have been reported in [25]. Experimental systems for force controlled robot drilling have been presented in [26], where a force controller with inner position control was used for the drilling thrust force control, and in [27], where an application to bone drilling in orthopedic surgery was presented.

While there exist commercially available products for force control from for instance ABB Robotics [28], none of the available packages include the particular features and/or level of flexibility required for the drilling application, i.e. the ability to redesign the inner-loop servo control for improved disturbance rejection.

Problem Formulation

The purpose of this work is a fully developed industrial prototype of robotic drilling, based on the use of high-performance force/torque control and light-weight industrial robots. The idea presented in this paper is based on applying a dynamically controlled pressure against the work piece with a tripod attached to the drilling tool, while a self feeding mechanism is used to feed the drill. This setup is as shown in Fig. 4.1. When used together with a metrology system for absolute accuracy in the initial positioning, the system should be able to satisfy the accuracy requirements of $\pm 0.2mm$, even in the presence of external load during clamp-up or drilling. The method of dynamic sensor-controlled drilling represents a different approach compared to current static (and expensive) systems. The purpose of the force control is threefold; i. to control the normality to the surface; ii. to avoid the drilling endeffector sliding on the surface (skating) during the drilling and clamp-up phases; and iii. to press the parts together so that burrs do not enter in between the plates. The control is accomplished by an open robot controller interface with a sampling rate of 250 [Hz] – see [29], [30]. Further, the system allows for reuse of much of the existing safety and programming functionality of the robot system, with extensions made to allow flexible programming of sensor-based tasks. The off-line programming environment DELMIA V5 Robotics was selected for simulation and planning of the drilling process, thereby simplifying robot programming. The robot hole-to-hole programming includes meta-data for the force control in pre-aligning the pressure foot. The force data is included in the robot program with our extension to the ABB Rapid program language called ExtRapid. The goal is to avoid additional force feedback programming after the off-line programming in DELMIA.

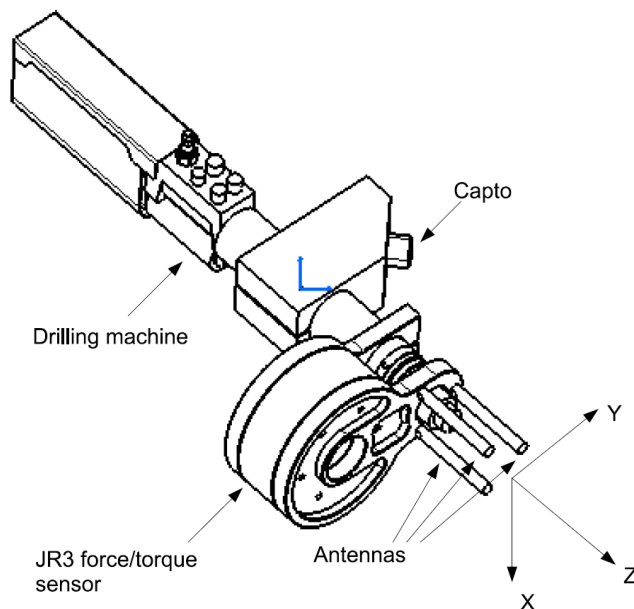


Figure 4.1 Overview of the end-effector prototype for the drilling experiments.

4.2 System Topology

The robot system includes a robot with its controller, two computers and a force/torque sensor. The solution presented in this paper has the sensor installed in the endeffector. Fig. 4.2 shows an overview of the system. In detail, the system includes:

- External computer: Master PC for ExtRapid program execution.
- Robot: ABB IRB 4400 Industrial robot.
- Robot Controller: Modified ABB S4CPlus control system.
- Power PC card: G4 processor with memory, PMC Interface.
- PMC-PCI card: Interface between the Power PC card and the PCI Backplane.
- Sensor: JR3/160M50 force sensor and corresponding computer interface.
- Network environment: Configured from the Master PC.

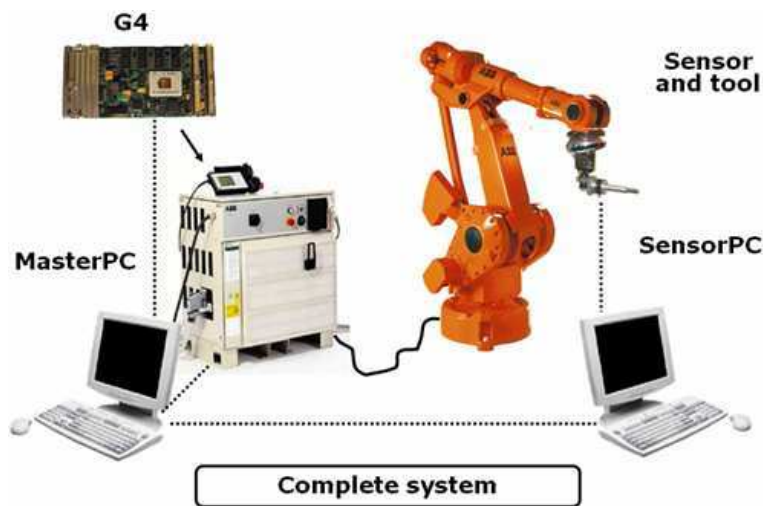


Figure 4.2 Overview of the robot control system. The sensor PC is an optional component not used in the drilling application, as support for JR3 force/torque sensors has been integrated into the robot controller system, and signals can be read directly into the Power PC card over the PCI bus. However, the ability to include a Sensor PC gives extra flexibility in interfacing other types of sensors typically used in laboratory environments, such as vision sensors for CPU-intensive real-time applications.

- Master PC Software environment: Making it able to execute and supervise ExtRapid programs.

The architecture of the ABB S4CPlus control system and its extensions are shown in Fig. 4.3. The force-control robot system consists of the robot and its controller, extended with an extra processor card (G4). The setup uses two additional PCs, MasterPC for executing the robot language extension (ExtRapid) and SensorPC for handling the force/torque sensor, responsibilities that are to be handled by the controller and the G4, but for development and testing purposes have been kept separate. The MasterPC has a double role, acting as cell controller.

Currently, the system is used in Lund, Linköping, and Valencia. Moreover, the platform was used for pre-product force control application development at ABB.

Robot force control programming is performed on three levels: (i.) Specification of a Simulink controller acting within the ABB industrial robot controller; (ii.) Steering of the controller through language extensions in the ABB robot programming language; (iii.) Task-level force annotations in the Delmia simulation software. The robot controller exten-

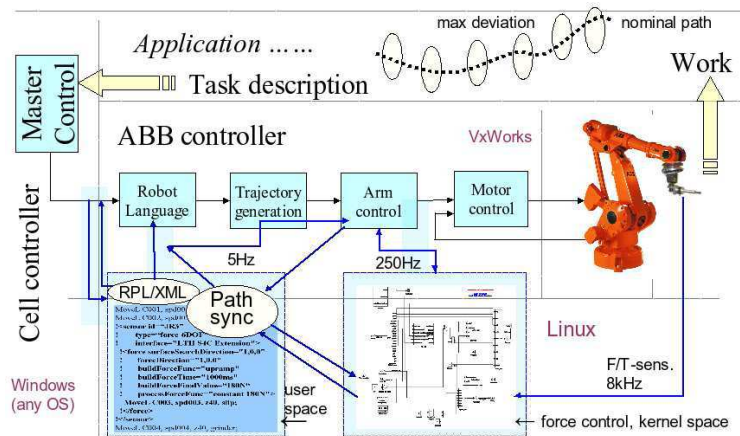


Figure 4.3 Extension of industrial robot controller with sensor interface and support for external computations and synchronization, using a Motorola PPC-G4 PrPMC-800 processor board mounted on a Alpha-Data PMC-to-PCI carrier board with a local PCI bus. Schematics overview from [30].

sion allows arbitrary Simulink models to affect the robot trajectory at the 4ms joint level. Developed models are compiled using Real-time Workshop and transferred to the G4 file system (NFS, FTP, ...) prior to execution. Execution time is not dependent upon ABB hardware but relies on the G4 processor, allowing for quite CPU-intensive models. Compiled models can easily be switched between task executions. A Simulink library offers blocks for reading and writing ABB trajectory data. External sensor data can be integrated as extra blocks, with data provided, for example, through the G4 Ethernet port. A low-level Java GUI (OpCom) allows direct access to the G4 for switching models, logging model parameter data and changing model parameters for tuning and validation in a running system.

At a higher level, model parameters are exported dynamically to the robot programming language extension (ExtRapid) where they can be accessed and modified as part of a regular robot program. Usage of a model is encapsulated in a controller block. Outside the controller block, regular ABB RAPID code is executed as usual. Inside the controller block, RAPID code is executed under the influence of the model. Since the model is not allowed to deviate too much from the planned trajectory for supervisory and safety reasons, RAPID code is needed for defining a default trajectory within the block. The interface between regular code and model influenced code is currently solved by requiring the robot system to be in a known state (non-moving, not-in-process) when turning on and off the model.

```
MoveJ pos10, v50, fine, drill_TCP
MoveL pos20, v20, fine, drill_TCP
FORCE
  FORCESET ramp_speed := 150;
  FORCESET glob_gain := 1;
  FORCESET f_switch := 1;
  WaitTime 0.1;
  FORCESET forceRef := -400;
  FORCEWAITUNTIL forceOut <= -390;
  WaitTime 1;
  SetDO bus2do8, 1;
  WaitTime 0.5;
  SetDO bus2do8, 0;
  FORCESET glob_gain := 0;
  WaitTime 8;
  FORCESET glob_gain := 1;
  FORCESET forceRef := 0;
  WaitTime 5;
  FORCESET f_switch := 0;
  WaitTime 1;
ENDFORCE
MoveL pos30, v50, fine, drill_TCP
```

Figure 4.4 The example force tag above defines a force scope in ExtRapid for *pos20*. The force is ramped up with the speed of 150 mm/s; *f_switch* command starts the impedance controller; *forceRef = -400* is the clamp-up force; *forceOut* is a trigger value to continue with the next ExtRapid program line; *SetDo bus2do8,1* activates the drilling cycle; *glob_gain = 0* stops the force control loop, which keep the robot steady during drilling. As *glob_gain = 1* and *forceRef = 0* is executed, the force control starts again and the clamp-up force is reduced to zero before moving the end-effector to *pos30* in the Rapid program; *pos30* is a position some distance away from the airframe.

The ExtRapid extension is implemented to be in conformity with standard RAPID on ABB S4 and IRC5 systems using compiler-compiler technology. Controller blocks are specified as add-ons to the grammar specification. As for programming, we refer to the example in Fig. 4.4.

This project used the off-line programming (OLP) method, with DELMIA V5 Robotics as the OLP system. The force-feedback programming method in this project was aimed at making the programming environment as powerful and user friendly as possible. In addition to the default installation of DELMIA a small interface was implemented to incorporate ExtRapid programming as part of the robotic drilling simulation process. In DELMIA there is a functionality to include comments to a robot program,

as the comment types PreComment and PostComment. The PreComment includes a comment prior to a program line in the post-processed robot program. PostComment does the same, but includes the comment text after the program line. For each position in the simulation model to be drilled the user clicks “Insert ExtRapid” and the ExtRapid menu is shown, see Fig. 4.5. Using PreComments and PostComments enables the user to view ExtRapid data directly in the PPR-tree. Using PreComment and PostComment was the key to avoid customizing the post processor in DELMIA, see Fig. 4.6 and Appendix. When the program is executed in the robot, the master controller interprets the ExtRapid data. One advantage having the master controller reading directly from the program in the robot controller, is the ability to include ExtRapid data in the teach-in programming method. The method to include the ExtRapid data in DELMIA made it possible to avoid the additional programming of force data on the workshop floor.

4.3 Drilling System Design

The end-effector prototype, an overview of which is shown in Fig. 4.1, was designed with the purpose of generating force and torque sensor data to the robot controller. The drilling tool was equipped with a tripod with three antennas positioned symmetrically around the drilling tip. When each antenna is pressed to the surface a force will build up and propagate to the force sensor. Three antennas were used to recognize asymmetrical forces around the drill tip in order to compensate for lack of normality. The Coromant Capto™ interface was used for attachment to the robot chuck. The force sensor was a JR3 160/50M, and was installed between the drilling unit and the pressure foot. The end-effector was configured as a mix between a pointing configuration and a hanging configuration [22]. The sensor data from the force/torque sensor was used in the following way:

- Torque: different amount of force on each antenna will cause errors in normality.
- Z-force: the total amount of pre-load force exerted on the surface.
- X-Y-force: the force indicating the skating effect.

All the six degrees of freedom (6DOF) of the force/torque sensor are used to compensate for the different phenomena in the process. The torque will increase if the end-effector is rotated around the X- or Y-axis (defined in the work piece plane), which corresponds to the end-effector force not

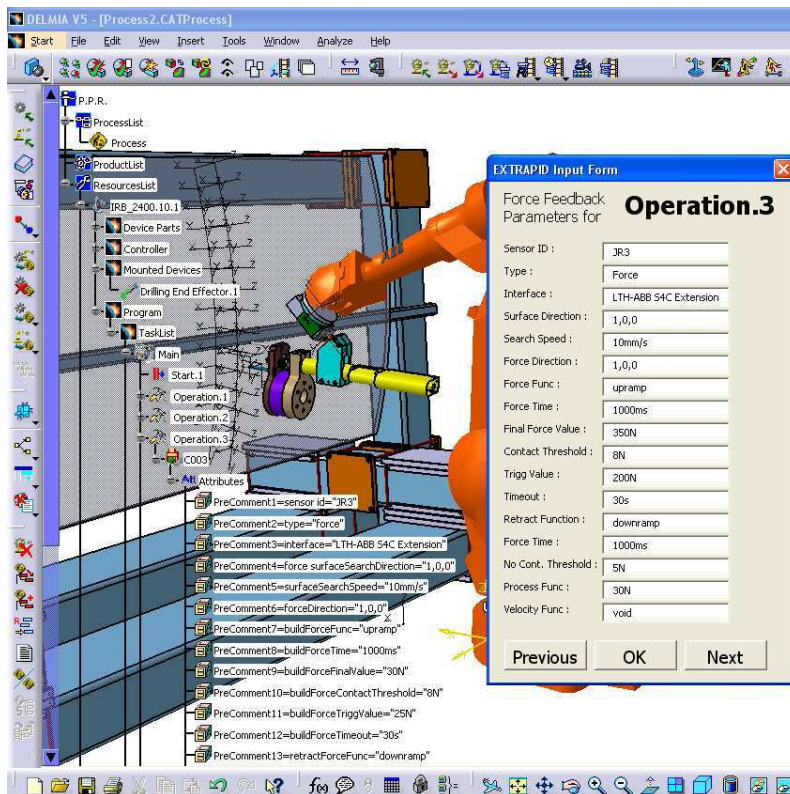


Figure 4.5 The Delmia interface with ExtRapid programming functionality.

being normal to the surface. Thus, the controller will make use of the torque measurements to even out the force on the three antennas. The controller will control the Z-force to ramp up the clamp-up force to the reference value and use the X- and Y force values to control the forces in the X- and Y-directions to zero, thus eliminating skating. Without compensation, due to the compliance in the robot transmission and links, deformations of up to several millimeters could occur, which would seriously degrade hole quality and positioning. Although the high-friction contact between the tripod and the surface improves the sliding suppression and effective stiffness of the robot, sliding may still occur if the tangential forces become larger than the break-away force of the friction contact. This is frequently the case in practice, as the axial cutting forces will make the tripod ascend from the surface, thus breaking the friction contact.

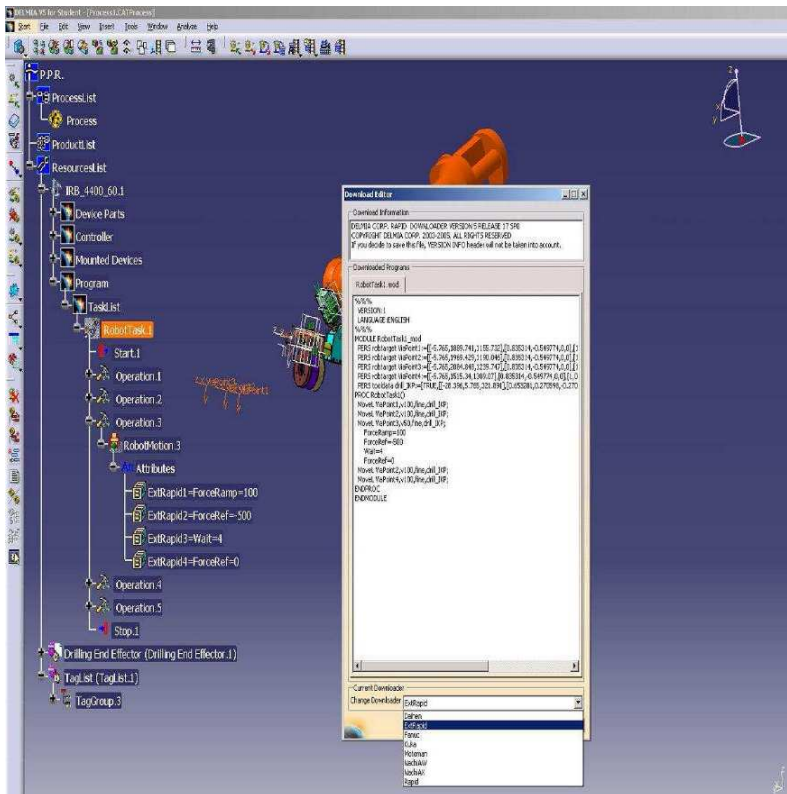


Figure 4.6 ExtRapid Task-level Programming in Delmia Software.

Dynamic Modeling and Control

For the purposes of simulation and model-based control design, a dynamic model of the system responses to external forces and motion references is required. The deflection in the tool position will not be in the direction of the external force. In particular, the axial forces on the drill and pressure foot will make the robot bend and deflect also tangentially to the surface, causing a sliding motion which must be compensated for. The presence of dry and viscous friction in motors and transmissions, mechanical backlash, and partially unknown parameters make it necessary to find a detailed model for high-bandwidth modelbased control design. Models capturing the behavior of the controlled robot were experimentally obtained according to [31], [32].

An extended approach including friction parameters was outlined but was not used in the final experiments, as friction was shown to be handled

more accurately on a lower level by the position/velocity control servos of the built-in controller. For this reason, the identified model used in the experiments was a linear MIMO system.

Environment Properties During stiction contact between the tripod and the drilled component, when the tangential forces became larger than the break-away forces of the stiction, the tripod started to slide across the surface, with poor hole quality and positioning as a result.

Therefore, it was important both to control the tangential forces so that sliding was avoided, and to control the moments to keep the tripod in contact with the surface at each of the three contact points.

For the tripod contact of the drilling tool used in this work, it was necessary to take also the geometry of the contact into account. The contact was considered as a combination of three-point contacts, where the force acting at each point contributed to the effective force and moment acting at the robot TCP point.

The developed contact model provided a useful local approximation during stiction, and the objective of the control was to keep the system in this stiction regime.

Control Design Because of the limited bandwidth of the motion control system and the deformations of the robot caused by external forces, the tracking of the desired motion may be poor when the robot is in contact with a stiff environment. In order to improve the tracking performance, compensation of the effects of the measured external forces was included in the inner-loop motion control. This can be seen as trying to increase the “stiffness” of the robot as seen from the tool, which improves the ability to control contact forces and moments. Although the compensation can only improve the rejection of disturbances up to a frequency limited by the mechanical and controller bandwidth, it is still effective as long as the variation of the sliding forces is slower than the bandwidth, which is usually the case in practice. We used a controller structure which includes this inner-loop compensation. The controller was chosen to give a proper suppression of disturbances at the arm side position for frequencies up to approximately 25% of the mechanical bandwidth, in order to suppress the dominant sliding effects.

In addition to force sensors, which can be used to obtain improved disturbance suppression through feedforward, feedback from arm side position measurements could be used in the inner controller to improve the absolute accuracy of the positioning. Such measurements could be obtained from, e.g., cameras or laser trackers.

Since the inner loop design was based on a 3-DOF model with translation only, a static decoupling matrix was included for improved decoupling

between the control of xy-torques and xy-forces. The use of a purely static method for the decoupling of the rotation can be motivated by the much lower demands on the bandwidth of the moment control, as compared to the linear forces. The reason for the difference is that the moment control only needs to keep the tripod aligned and in stable contact with the surface, and is not as strongly affected by the fast variation of the forces that cause the deflection and sliding. A proper choice for decoupling matrix could be found from the static calibration data, as described in [31], [33], [34], [35].

4.4 Experiments

The first part of the experiments were carried out at Lund University, using a smaller ABB Irb 2400 industrial robot equipped with a pneumatic Atlas Copco LBL25 drilling machine with 4 mm drill diameter. The contact forces were measured using a JR3 force/torque sensor, and the workpiece was a 3.5 mm plate of high-strength aluminum. In this part of the experimental program, a number of experiments were performed in several different robot configurations, with the purpose of evaluating the effect of the sliding suppression control. The evaluation was done by comparing the results from two different sets of experiments, corresponding to controllers with and without sliding suppression, respectively.

In Fig. 4.7 it can be seen that the tool deflection when forces were applied during the force buildup was reduced to approximately 0.1 mm, and sliding during the drilling phase was reduced to 0.1 mm. Having performed a number of experiments in different configurations, the model-based force controller was always able to control the sliding forces so that the tripod contact remained in the stiction regime during the entire drilling operation, and the tangential deformation was always below 0.3 mm. This resulted in greatly improved mechanical stiffness and vibration suppression, leading to significant improvements in hole quality and positioning.

A second, more rigorous, test program was performed in the robot lab at Linköping University. Test coupons were installed in a test rig at four different locations in the robot work envelope. In each experiment the skating deviation was measured using two LVDT sensors (Linear Variable Differential Transducer). The test program included four stations and on each station three tests were made:

1. to press with different clamp-up forces on the surface using a hard foundation;

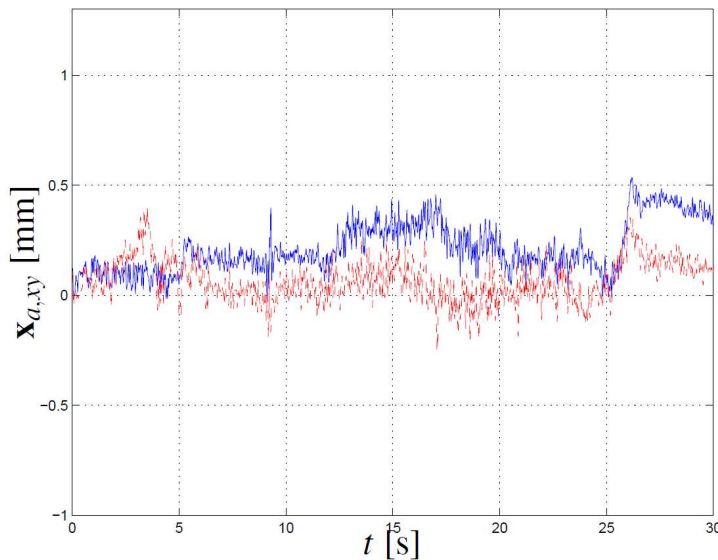


Figure 4.7 The linear deflection of the tool in the x- and y-directions during a drilling experiment using an inner-loop controller with compensation for the robot compliance, and with active control of the sliding forces. The drill sliding is reduced in the critical drilling phase by a factor of five as compared to the previous case.

2. to press with different clamp-up forces on the surface using an elastic foundation;
3. to keep the clamp-up force constant and drill.

with measured outputs

1. forces and torques in all six degrees of freedom;
2. skating using LVDT sensors.

The elastic foundation constituted four rubber pillows and two square shaped plates. When the robot pushed the test coupon intentionally moved. The idea of a moving test coupon was to simulate flexible skin panels. In this case the LVDT moved along with the mobile test coupon. The outputs of the tests were the measured forces and torques in all six-degrees-of-freedom, and the skating as measured using the LVDT sensors.

In order to measure the skating during clamp-up, the drill bit was replaced with a short rod having square cross-section. This solution enabled the LVDT to measure skating as close to the drill tip as possible.

The robot was moved close to the surface before the measurements were initiated. This small movement before build-up of the clamp-up pressure caused a small movement, which was neglected when measuring the actual skating effect. The force feedback control was initiated as soon as the force was detected in the force sensor. The test results showed an LVDT motion of 0.1 mm. It was concluded that most of the 0.1 mm movement was due to the orientation control of the end-effector. The reason for this was that the LVDT could not measure exactly at the TCP point, but rather at a point 3-4 mm up along the square shaped stick. This phenomenon was increased when drilling on flexible surfaces, where larger orientation control actions were necessary.

The drilling was performed with a Gühring cutter with a 5 mm bore diameter, and the holes were drilled 15 mm apart. The hole-to-hole cycle time was around 12 seconds. The actual time to control the clamp-up without skating and with the end-effector orthogonal to the surface took around 2.5 seconds. In addition to the coupon drilling tests drilling was also made on a very flexible plate. This test was made to investigate how well suited this method is to be used in flexible airframe skin panels, as seen in Fig. 4.9. In the cases where the structure has low stability, the normal direction of the skin will change during clamp-up. This is one of the strengths of using force/torque feedback. In this case the end-effector maintains orthogonality with respect to the surface during clamp-up, through rapid rotation of the tool in order to adapt to the changing surface normal. This was confirmed in laboratory experiments, drilling on the very flexible skin panel.

4.5 Discussion

As an alternative to controlling the position using arm side position feedback, the controller attempts to achieve sliding suppression by making sure that the tangential interaction forces are always small enough to keep the contact in the stiction regime. In practice, the achievable bandwidth of the force control is limited by the mechanical bandwidth of the robot, as well as by the bandwidth of the inner motion control. Therefore, the force control and sliding suppression must be combined with proper mechanical construction in order to obtain a sufficiently stiff system. In the proposed solution, the stiffness is effectively increased by the tripod high-friction contact. The contact will damp and suppress small disturbances, such as vibrations from the feeding and rotation of the drilling tool, while the force control and active sliding suppression handles large disturbances at lower frequencies, such as the slower variations of the cutting forces. Thereby, a system which is able to reject disturbances over

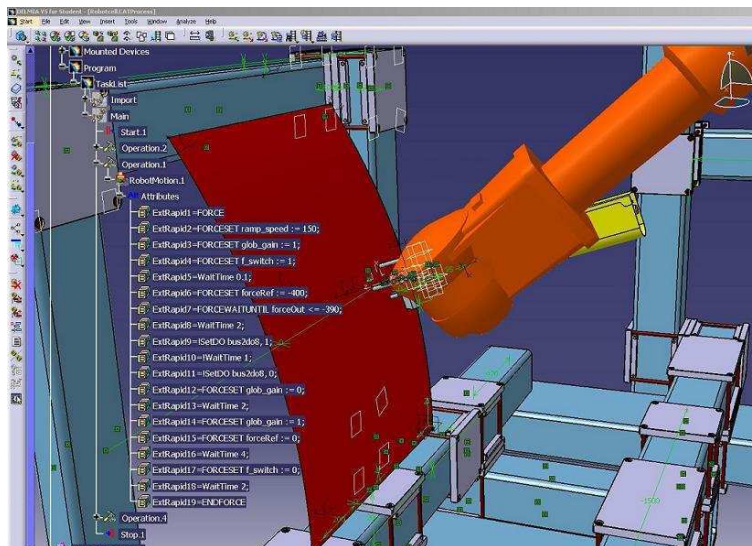


Figure 4.8 Task-level program set-up for testing of drilling in flexible airframe skin panels.



Figure 4.9 Set-up for testing of drilling in flexible airframe skin panels according to task-level program of Fig. 4.8.

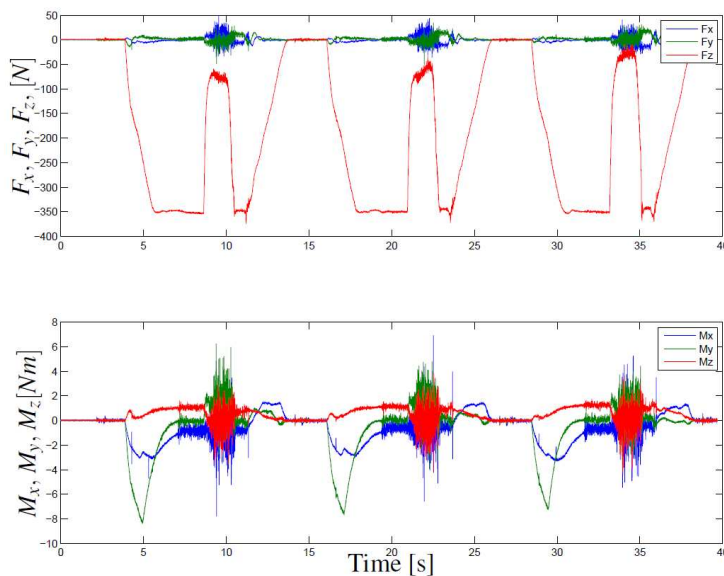


Figure 4.10 Drilling operation forces F_x , F_y , F_z , M_x , M_y , M_z vs. time for the end-effector with three antennas also reflecting the drilling operation cycle time, the plate thickness being 4mm with holes at a distance of 15mm.

a wide frequency range is obtained, at a potentially very low cost as the solution requires only a standard force/torque sensor integrated into the drilling tool. Whereas there are difficulties with coupling between different directions in the Cartesian space and the linear-model approximation could be challenged, it should be stressed that in spite of these approximations and error sources it was possible to obtain the performance needed for the process.

In addition to the robotic force control interface [29], [30], robotic force control interfaces and their usage have been reported elsewhere – e.g., the DLR effort [36], the Comau force control interface [37], and the ABB force-controlled machining [38].

4.6 Conclusions

The use of industrial robots in automatic drilling applications has been limited, mainly due to the presence of rapidly varying interaction forces in combination with compliance in gear boxes and links. Functionality for high-bandwidth force control in modern industrial robot control sys-

tems could potentially lead to robotic drilling systems with significantly improved performance, without the use of costly hardware modifications and calibration procedures. In this paper, we have presented methods and systems for force controlled robot drilling. Using a 6-DOF force/torque sensor, an outer force control loop and a model-based inner-loop disturbance compensation scheme have been designed, and used to control the axial contact force and suppress the sliding of a tripod contact while the drilling is performed. The advantage of the proposed controller is demonstrated in reproducible drilling experiments using an industrial robot system. Moreover, the application potential for high-precision robotic drilling operations in airframe assembly has been demonstrated.

Part II

**Configurable Modular
Equipment**

5

Configuration Support for Parallel Manipulators

Goal: Software support for configuration of modular robots

5.1 Introduction

New low-cost and flexible robot concepts are needed to fulfill the needs for small lot production series, typically found in small- and medium-sized enterprises (SMEs) in manufacturing; SMEs depend on their ability to cost efficiently produce customized products, and the use of manual labor is common to accomplish the required flexibility. To maintain profitability on a global market, there is a desire to have robots that in an efficient way can assist human workers. This would require robots to be much more flexible to configure and use, and in many cases much more stiff in the sense of motion compliance compared to traditional industrial robot arms. The Parallel Kinematic Manipulator (PKM) with Gantry-Tau structure has been proposed recently to accomplish a low-cost reconfigurable stiff robot [39]. The Gantry solution provides a larger working range than other PKMs do. The Tau variant provides an open and accessible work space, which is an important advantage over, e.g., the linear Delta version. Opposed to traditional serial/articulated robots that come as an optimized fixed unit from the robot manufacturer, a Gantry-Tau PKM can be assembled at the customer site and reconfigured to meet various task requirements, without losing stiffness or speed if reconfigurations are within reasonable limits. However, reconfiguration of the robot demands on-site calibration to guarantee absolute accuracy of the manipulator. Figs. 5.1, 5.2 show two early LTH prototype Gantry-Tau of varying sizes. In Fig. 5.3, we see two later prototypes of the Gantry-Tau, developed within the EC FP6 SMErobot project [40]. The upper left picture in Fig. 5.3 shows

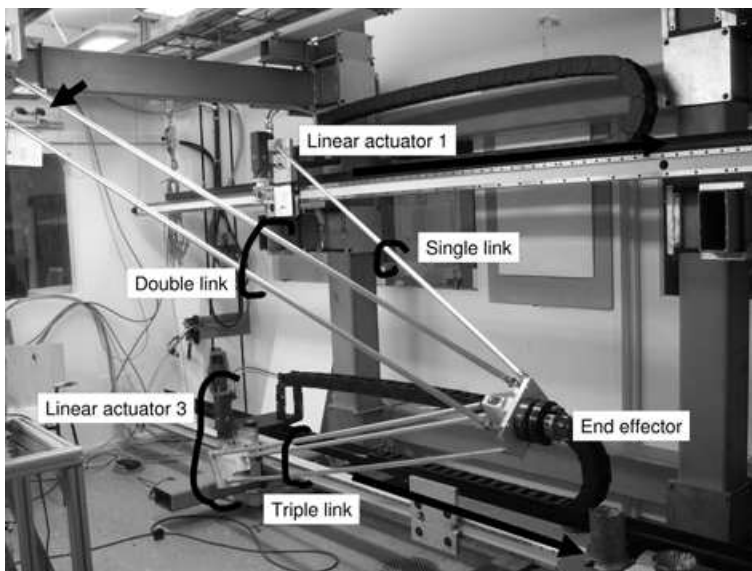


Figure 5.1 Prototype 3-DOF parallel robot with Gantry-Tau structure built using a modular framework carrying Güdel linear actuators. The links shown in the picture are controlled through an ABB IRC5 industrial robot controller. Note that the links are not dimensioned for industrial usage. The end effector carries devices for collision protection and tool exchange (cabling for the wrist not included in picture). To the upper left can be seen cameras to be used for signature calibration.

the robot integrated with an industrial robot controller where it is used for contact force controlled grinding and cutting applications within the foundry industry. The upper right picture shows a vertical prototype with dual motor control for backlash reduction and improved performance. The robots are modular in their design and the base construction consists of three linear actuators working in parallel with arm links connected to the robot wrist using the Tau structure. Both the robot and the framework carrying the robot are built using modular components to ease fast deployment. The framework in the upper right picture is based on the box-joint system designed at Linköping University [41].

Arm lengths can easily be changed by replacing arm links, and the linear actuators can be moved in space by adjusting the framework, thus easing reconfiguration. The overall design and kinematic configuration needed when assembling at a customer site is related to the engineering that has traditionally been done at the robot manufacturer site, but with the increased modularity and the mechanically not redundant link system there is now an opportunity to base the design less on the limits of robots



Figure 5.2 Table-sized Gantry-Tau prototype shown at the Scandinavian Technical Fair (Stockholm, 3-6 October, 2006). End effector, linear tracks, and gearboxes manufactured inhouse. Drive systems from the Faulhaber Group [6]. Robot controlled from the Visual Components 3DCreate tool (laptop).

from robot manufacturers, and more on the requirements of the end-user, which may shift over time and application. A key property for the end-user is the usage of a standard industrial robot controller whereas an open R&D-platform is to prefer during the development phase. In a series of previous research projects on contact-force control, an open robot control platform for fast sensor interface was developed by Lund University and ABB [30, 43]. Furthermore, the SMERobot partner Güdel AG [42] has experience in integrating their modular robotics components with ABB robot controllers. Therefore, an ABB IRC5 controller was chosen together with Güdel linear actuators to form the core of the prototype. Then, in cooperation with ABB, PKM kinematics were developed and integrated into the (standard) IRC5 system. In parallel, there is a R&D platform based on the IRC5 controller for further algorithm development.

It is important to support the new configurability and modularity in robot simulation and programming tools. For end user testing, a PKM work cell is built at the Castings Technology International, Sheffield, to support a medium-sized foundry company (Norton Cast Products Ltd [44]). The applications are mainly for cutting and grinding operations of foundry goods. In the early design phase, simulating the process is neces-

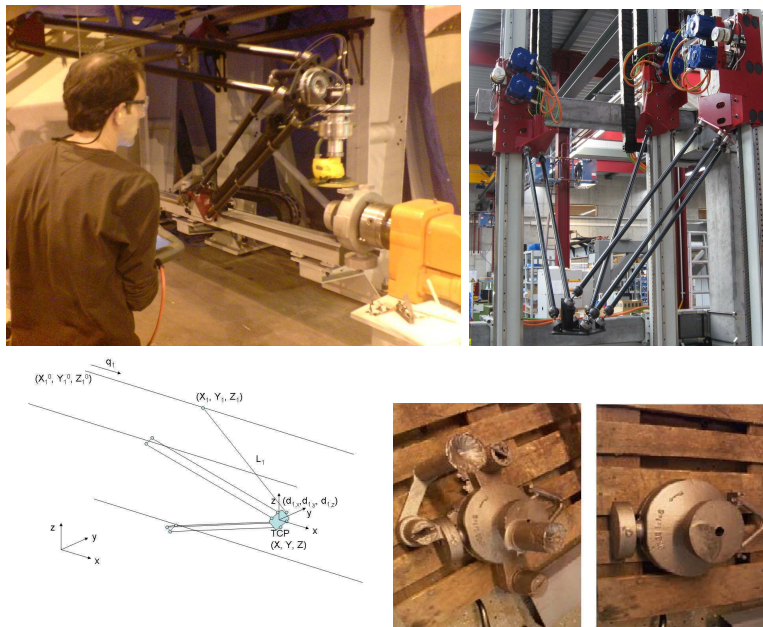


Figure 5.3 Parallel kinematic robot in force controlled cutting operation on a casted pump house at CTI, Sheffield (upper left); Vertical Gantry-Tau PKM at Güdel, AG, Switzerland, with dual motor control [42] (upper right); Schematic picture of PKM and its links (lower left); The pump houses (or work pieces) before and after (manual) cutting (lower right).

sary in order to dimension the PKM for the task. A configurable Gantry-Tau simulator was developed for a robot tool that targets end-users in the work-cell planning phase, and together with tools necessary for configuring the kinematic structure of the robot at this phase it provided the necessities for the initial planning in the foundry work-cell scenario.

The rest of the paper is organized as follows: Firstly, issues regarding PKM kinematics and its application in a controller are discussed. Then the support in end-user tools is discussed and the concept *task-centric* reconfiguration is defined. The article concludes with further perspectives and future work.

5.2 Gantry-Tau Kinematics

The basic design of the Gantry-Tau robot (Fig. 5.3) consists of three parallel prismatic joints moving carts on linear tracks [42]. The three carts

are connected via links to a wrist mount. The six links are clustered into three groups (arms), with one, two and three links, forming the Tau structure. The links are connected to the cart and wrist mount through passive spherical joints. During the project, new spherical joints with very high stiffness in relation to weight and low friction were developed using bearing structures with diamond-like carbon layers. The links belonging to the same cluster form parallelograms.

3-DOF Kinematics Gantry-Tau kinematics for 3 degrees of freedom was implemented both in the standard ABB IRC5 controller, in the above mentioned R&D platform, and in PC control for the Güdel-based vertical Gantry-Tau PKM (Fig. 5.3) equipped with Beckhoff drives [45].

The Gantry-Tau kinematic solution for 3 degrees of freedom is described in [46]. In contrast to the case for serial manipulators, the forward kinematics problem is the more difficult problem to solve for PKMs. For an analytical solution the 3-DOF forward kinematics solution assumes parallel prismatic actuated joints and a fixed orientation of the wrist (which is guaranteed by proper placement of the links in the Tau structure). The problem can then be reduced to a stepwise geometric solution where first the intersection of two link clusters is calculated and then the resulting circle is intersected with the third link cluster. For the forward kinematics, two solutions exist and a configuration state is needed to decide which one is valid. The forward and inverse kinematics solution for the three-base actuated prismatic joints can thus be represented as

$$q_x = f_3^{-1}(x, s, c) \quad (5.1)$$

$$x = f_3(q_x, s, c) \quad (5.2)$$

where the joint positions $q_x = (q_1, q_2, q_3)$ are related to a wrist center point (WCP), position $x = (x, y, z)$, and parameterized by a configuration state (solution selection) c , and a structural parameterization s . As the configuration state resulting from direct kinematics is not related to the inverse kinematics configuration, c is defined to include both kind of configuration states. The kinematics solutions have been implemented in Matlab and C for control purposes, and in Maple and Python [47] for simulation and analysis purposes. For the R&D controller platform, the kinematics are represented in Simulink/Real-time Workshop blocks for fast and easy control and application development (i.e., no proprietary controller code needs to be changed in modifications of control algorithms or the kinematic implementations during a development phase).

Kinematic descriptions

To capture and support the modularity and flexibility of the Gantry-Tau PKM as needed for manufacturing SMEs there is a need for a common higher-level kinematic description that can ease implementation of new software tools, methods, and algorithms. Symbolic representation of kinematics is necessary for developing methods for calibration and configuration such as identifying geometrical and dynamical properties of an assembled robot. As an example, a higher-level kinematic description would considerably ease integration efforts between applications. One experiment would aim towards integration of two applications featuring discrete-event simulation of workcell layouts (3DCreate [48]) with automatic program generation (RinasWeld [49]). A common higher-level kinematics description enabling the two software environments to share robot geometry and kinematics would considerably ease the integration effort. To this end, the Modelica language was tested by using it for kinematic analysis, Modelica being an object-oriented modeling language [13]. Dymola is a commercial implementation of Modelica, which is used in this work (Dynasim [15]). Figure 5.4 shows a Modelica model of the Gantry-Tau robot. Two different kind of models have been implemented to model direct as well as inverse kinematic behaviour. The Modelica Multi-Body Library models kinematics and dynamics (kinetics) of rigid body systems. An advantage of Modelica is that mechanical models can easily be extended with models from other domains, e.g., a controller or an actuator for the PKM. Together with additional files provided by Dynasim, C-files generated for simulation can be used for hardware-in-the-loop simulations, e.g., for control of the robot. This, together with other experiences from manual implementation of a GT kinematic solution in C for ABB IRC5, implementation in Maple with code generation towards Visual Component specific Python, and modeling and analysis in Modelica points clearly towards a need for a common kinematics description valid across both controllers and tools.

Higher degrees of freedom

The Gantry-Tau has a natural modularity in extending the degrees of freedom. Several different ways of extending the existing 3-DOF robot to 5-DOF are conceivable. First, mounting an active wrist with two rotational DOFs on the existing wrist mount were considered. A second possibility was to add two supplementary carts on two of the tracks and have the six links distributed on five link clusters. A third is to add rotational joints on two of the carts to transfer re-orientations to the wrist mount through the link clusters, as done in Fig. 5.3 (upper left).

For the first case, an analytic solution of the kinematics problem is

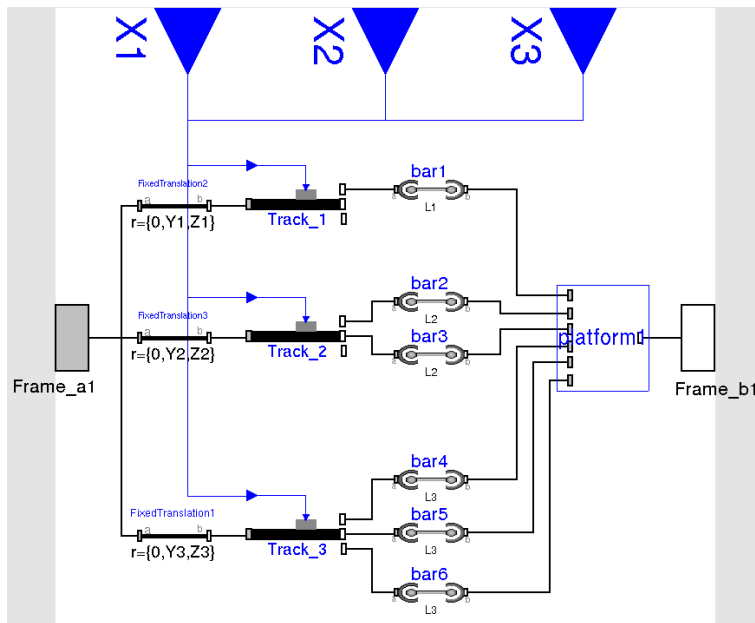


Figure 5.4 The Modelica language supports analytic handling of kinematic loops, which is an obstacle in many other simulation and modeling tools. The multi-body library is used for modeling the kinematics and dynamics of a 3 DOF Gantry Tau-PKM.

easy to derive. In this case, position and orientation can be regarded separately. For forward kinematics, first the platform position X_p is calculated from the joint positions [46]. Then, the WCP position X and orientation R are obtained by considering only the active wrist with rotation angles $q_\theta = (\theta_1, \theta_2)$ and kinematic parameterization s_w (e.g., Denavit-Hartenberg parameters), and $f_2(q_\theta, X_p, s_w)$ calculated [50]:

$$(X, R) = f_5(q_x, q_\theta, s, s_w, c) = f_2(q_\theta, f_3(q_x, s, c), s_w) \quad (5.3)$$

The inverse kinematics problem can be solved similarly.

For the foundry application, a compact wrist with new lightweight servo actuators including new harmonic drive speed reducer and new motors from HDD (servo actuator weight 2 kg, outgoing static torque 80 Nm) was developed (Fig. 5.5). Moving link 4, a pure rotation around the z-axis can be performed (Fig. 5.7). Moving link 2, 3, 5 or 6 separately gives a composed rotation around all axes (Fig. 5.7 for link 2). Simulation experiences show that rotation limits exist which may vary considerably throughout the work-space. Depending on the TCP position and the con-



Figure 5.5 The 2-DOF serial wrist with two lightweight actuators including high-power HDD-motors mounted on the (black) “wrist frame plate”. A wrist-mounted force/torque sensor (ATI Omega160) is used for lead-through programming and contact force controlled cutting/grinding.

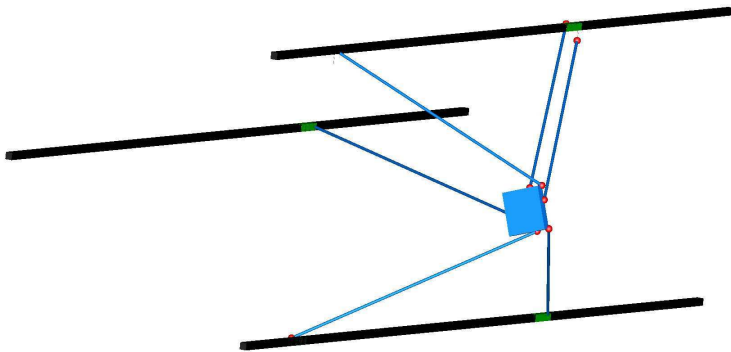


Figure 5.6 Extension to 5-DOF with two additional carts.

sidered link, the angular limits arising from moving a single link have values from $\pm 3^\circ$ to $\pm 103^\circ$. These values have been obtained without considering possible angular limits of the spherical joints [51].

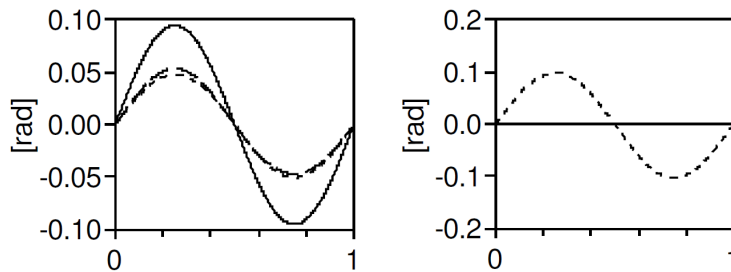


Figure 5.7 Simulation results: TCP orientation results from moving link 2 (left) and link 4 (right), using successive rotations around x- (solid), y- (dashed) and z-axis (dotted).

5.3 Gantry-Tau Configuration Support

The modular design of the Gantry-Tau PKM allows part of the overall design and kinematic configuration traditionally performed at a robot manufacturer site to be exposed as configuration options towards the end-user, thus increasing the robot flexibility towards the end-user application. However, (re)configuration options must be analysed and handled already in the work-cell planning phase for the end-user to benefit from these properties optimally. This calls for reconfigurable simulation models to be included in robot tools together with robotspecific tools for analysing/deriving robot configuration properties (pre-calculated for traditional non/limited-configurable robots and normally available from the robot manufacturer), such as work-space envelope, for comparison to task requirements, and (eventually) synthesizing configurations based on robot-task requirements.

Automated calibration routines

Every physical reconfiguration of the robot frame or change of link lengths requires a succeeding recalibration to guarantee the accuracy of the robot. In general, signature calibration of robots are done at the robot manufacturer using high-precision metrology equipment. For a reconfigurable manipulator, the need for a low-cost, high precision, automated calibration method is needed.

In [52], a method for fully automatic kinematic calibration of a robot using two cameras was presented. This method includes calculation of appropriate measurement points, automatic experiment execution, automatic pattern detection and localization to determine the robot pose and identification of the kinematic model by model-error minimization. Estimation of the kinematic model's positioning error resulting from mea-

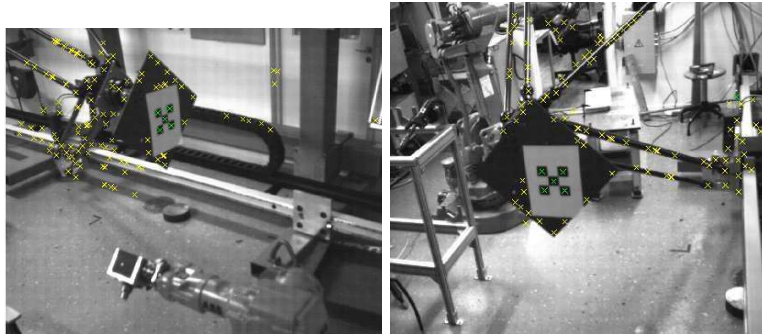


Figure 5.8 The automated calibration routine in [52] repositions the robot and the wrist mounted calibration pattern on a semioptimized grid in the robot workspace (based on approximative parameter values). Accurate kinematic robot parameters are then calculated based on measurements of the corresponding stereo pictures and the robot joints values.

surement and modeling errors is given. The method achieves subpixel resolution for the pattern positioning, Fig. 5.8 showing a pair of stereo pictures of the robot with calibration pattern.

Task-centric configuration in foundry scenario

Manufacturing in the traditional foundry industry comprises several challenges for a flexible automation, including small-lot series and individual variations in castings. Due to harsh working conditions, cutting and deburring in foundry industries is an important robot application. The foundry scenario was specified from an end-user view, based on typical applications for one of the SME project partners (Norton Cast Products Ltd) [44], with a work cell set up at the Castings Technology International (CTI), a provider of the new technique and application [53]. Foundry workpieces came in small batches (estimated 1-10), varying sizes (estimated 10-1000kg), and with varying geometries. The application consisted of material removal by cutting input metal reservoirs, which are needed for the casting process but which are not part of the product (cf. lower right pictures in Fig. 5.3). The cutting process is followed by stub grinding and for fins also deflashing with cutting forces depicted in Fig. 5.11. In the initial planning stage, a feasible task family was chosen and appropriate robots and external axes were selected for the tasks. For a configurable PKM, this stage also involved selection of appropriate structural configuration parameters for task-fitting the robot geometrically, in this case cutting of low-weight work pieces (10-100kg). This set-up was one of the SMErobot key demonstrators for new easy-to-use robot concepts [40].

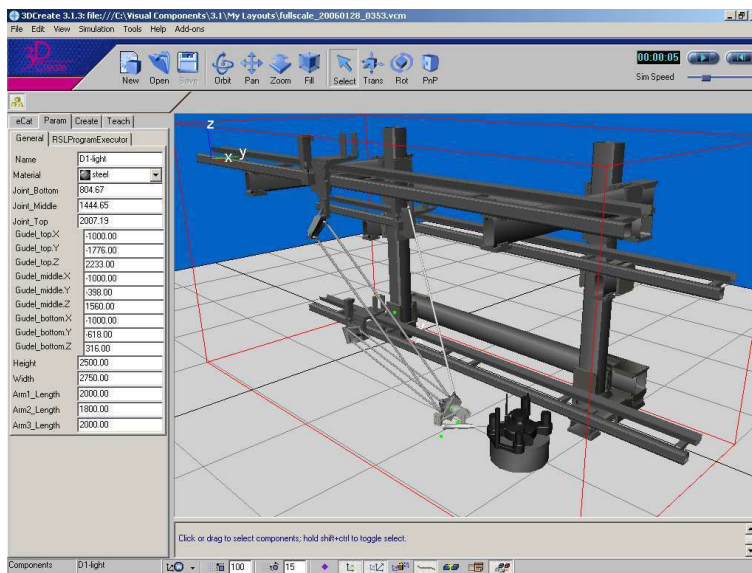


Figure 5.9 Prototype SMERobot foundry workcell using a reconfigurable Gantry-Tau PKM for material removal. To the left are reconfiguration parameters for the modular framework supporting the PKM.

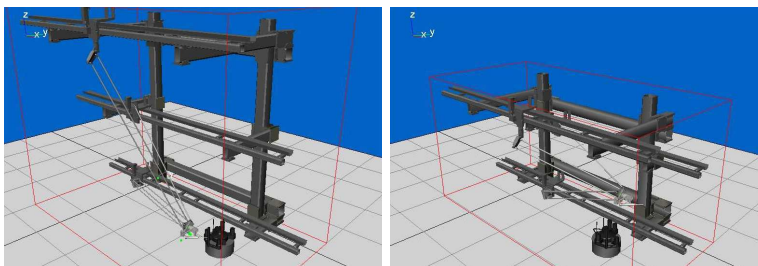


Figure 5.10 Gantry-Tau before (left) and after (right) re-configuration.

Properties considered interesting in the initial work-cell planning stage were work-space envelope (to handle varying workpiece dimensions), reachability of cutting tool towards workpiece, dexterity (for good cutting process), determination of necessary degrees of freedom for wrist (to reduce weight), and determination of the necessity of an external axis for the fixture (to increase reachability). Configuration decisions were to be assisted by simulations performed in a work-cell and factory planning tool. To configure the robot to better fit the task in consideration, a reconfigurable robot simulation was implemented in the 3DCreate tool [48]. For

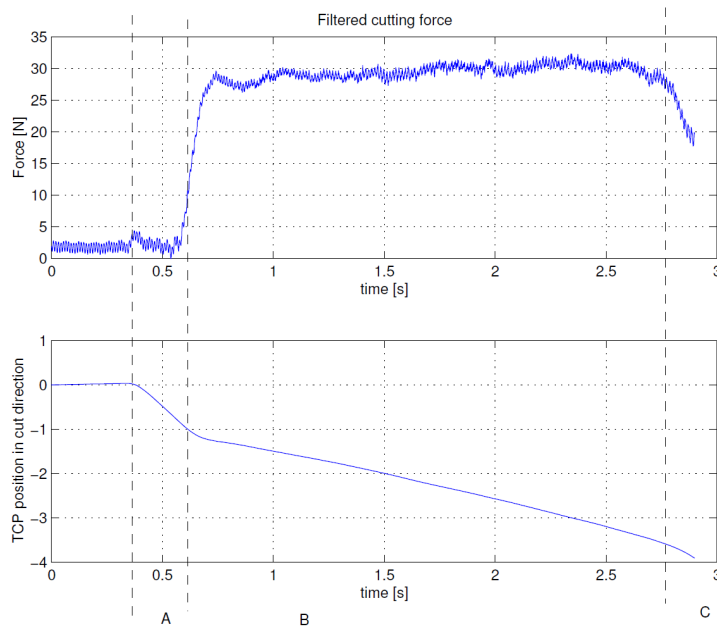


Figure 5.11 Cutting force from slitting operation (upper). Corresponding TCP-position: Approach phase (A), cutting phase (B), after cutting through (C) (lower).

the initial planning a family of work-cell simulations with the robot were then created featuring different wrists (1, 2, 3 DOFs), different fixtures, and different number of external axes (0, 1 DOF). The size of the robot was selected by reconfiguring the robot with ranges of link lengths and track positions to find a good working envelope for the task while adhering to volumetric constraints (room size). Reachability and collisionfreeness were confirmed for the example workpiece CAD model by explicitly programming and simulating the cutting process for promising work-cell candidates. This also gave an indication of how the work-cell might perform for other workpiece geometries, although further testing would be necessary. Figs. 5.9 and 5.10 show one candidate work-cell with robot reconfiguration, selection of wrist and fixture, and result from work-space envelope analysis of one robot configuration (easily covering the work-piece).

Virtual tools to ease task-specific configuration analysis

To assist in the initial work-cell planning stage it was important to ease configuration analysis in the simulation tool. In particular, easily accessi-

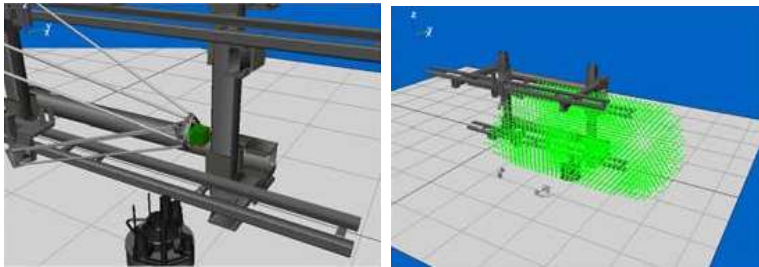


Figure 5.12 Work-space exploration end effector mounted on Gantry-Tau robot (left). The tool implements a simple grid-based algorithm to estimate work-space envelope. Outcome of the tool (right).

ble methods for deriving robot configuration properties were needed. Later a need for task knowledge in the simulation tool was discovered.

Availability of automatic programming for the cutting process would probably have meant significantly reduced time spent on verifying reachability and collision-freeness at the initial planning stage. Having these needs, and at the same time considering the component model used in the simulation tool, the solution was to create virtual end effector tools containing both tool geometry and process/task knowledge, thereby allowing the tool to automatically program any robot that it was mounted on. Two examples of virtual tools were created. Fig. 5.12 shows a pure virtual end-effector tool (no representation in the real world) used for obtaining an estimate of the robot work-space envelope for a given configuration. It contains a parameterized robot task (search volume, sample step) that defines a discrete volume grid search algorithm noting reachability for each visited position in space. The result is stored as a program in the robot containing all reachable points, and may be viewed by the simulation tool. Fig. 5.13 shows a touch sensor virtual end effector tool. The tool contains a grid search algorithm to probe the surface of a workpiece placed in front of the robot which can be used, for instance, for analysing reachability and collision-freeness towards a CAD workpiece for given configurations. Unlike the previous tool, the virtual touch sensor may present a real tool, and might even be used in path planning the search for the real workcell. Eventually, in this particular case, the virtual tool geometry is to be replaced with the geometry of a real touch sensor and moved to the full-scale Gantry-Tau robot. Both tools were created following the component model of the simulation tool and were implemented using the builtin script language (Python [47]).

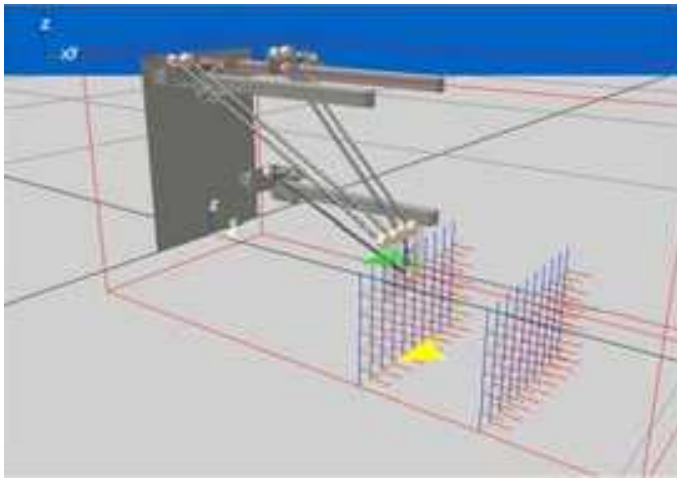


Figure 5.13 Desktop version of Gantry-Tau equipped with a fictitious touch sensor and programmed with a search pattern.

5.4 Conclusions

The reconfigurable Gantry-Tau PKM is a new robot concept that has the potential to achieve a cost/performance ratio that would make it highly attractive for SME manufacturing. To support reconfiguration, it was experienced that robot simulation and programming tools need to provide aid in terms of reconfigurable robot models and end-user analysis methods. For implementation, 3DCreate provided good support through its component model, plug-and-play capability and the flexible Python-scriptable engine. To increase flexibility, two possible extensions of existing 3-DOF kinematics to 5-DOF were presented.

To assist engineering analysis and synthesis of the robot workcell, a task-centric view was proposed to ease adaptation of the robot configuration towards the application. As an example, the work-space analysis tool (implemented in Python as a virtual end-effector) turned out to be easy to use by other users, knowing nothing about the internals. For robot programming and task configuration in general, a non-robot-centric view was proposed, where task and process knowledge are associated with the end-effector. The end-effector also programs the robot, so that by mounting a cutting tool the robot effectively becomes a cutter and is configurable for cutting operations. The modularity and flexibility of the GT-PKM, together with the needed toolkits for configuration and task definition, have made us realize that there is a need for a common higher-level kinematic

description that can ease implementation of new methods and algorithms. Code generation towards explicit controllers, low-level code, robot models, and other representations was equally important to lower the amount of engineering time needed. Symbolic representation of kinematics and other properties are necessary for developing methods for calibration and configuration such as identifying geometrical and dynamical properties of an assembled robot. Our conclusion is that such an increased level of abstraction is of key importance to fully exploit the increased flexibility, without too extensive engineering efforts.



Part III

Human-Robot Interfaces

6

On the scalability of visualization in manufacturing

Engineering tool in user interface

6.1 Introduction

Virtual manufacturing environments are emerging to meet the demands of rapid planning and reconfiguration in production systems. That in turn is driven by the need to rapidly respond to product changes, thereby achieving shorter time to market and increased profitability.

The similarities between factory automation and both virtual reality and enterprise computing have been observed [54]. There are, however, also important differences. In the world of factory automation, as well as in other areas such as autonomous systems etc., we have to cope with the differences between the real and the virtual world, not only during creation but also frequently during tuning, operation, and maintenance of the system. This imposes additional requirements on the user interaction, which may take place first in an engineering environment and later in a production environment (using the same software component). Furthermore, software components may be used for, or tightly together with, the control of physical machines. In order for this to scale up to large, as well as down to small systems, we will first have to find a sound software technology. Secondly, we need an appropriate technique for supporting graphical user interfaces. As the most challenging case, we focus on the need for 3D graphics, which clearly applies to many programmable machines such as industrial robots.

We propose the notion of executable visualization graphics as a term for the encapsulation of graphics and renderer together in a software component. The immediate advantage is that one may use a high-level graphical description language but still be able to render on systems providing only low-level rendering capabilities. Other advantages are customized navigation and animation capabilities, easy management and customization of graphical descriptions, possibility to use template software components for creation of a whole class of visualizations, and construction of visualization components for heterogeneous environments. We believe this approach will be highly useful when dealing with small application specific visualizations.

After presenting some related approaches, we will first look at the issue of scalability and its implications for manufacturing software. That points out some unsolved issues concerning software components and graphics. The main part of the paper is the subject of executable visualizations. A prototype implementation using Java technology is presented followed by a discussion with application examples. Finally, conclusions are drawn.

6.2 Related work

Web visualization is a field which is in the beginning of its development. This work has largely been inspired by the efforts of Dr. Mikael Jern to create componentbased web visualizations [55]. Visualizing medical data on the web is a problem because most medical examinations produce huge amounts of data. The low bandwidth of Internet produces the need for data reduction techniques reducing the amount of data being sent to the web visualization client. Dr. Jern is considering client-server solutions based on intelligent component clients containing rendering, custom navigation, and visualization functionality to assist the user in searching the dataspace while minimizing the data transfer rate.

The German Institute of Space Flight lead by Dr. Hirzinger has made an early attempt to use web-based visualization for telepresence [56]. Their group has developed virtual robots used for online prediction of robot movements in teleoperating environments.

Most notable within manufacturing visualization is the field of digital manufacturing, creating large scale visualizations covering entire plants. The production shall be suited to the product: the goal of digital manufacturing is to connect manufacturing control to the product planning process. By simulating the entire production process in a virtual environment shorter product cycle times are achieved, as well as lowered costs, guaranteed quality and shorter product-to-production time spans [57].

Robot visualization is being used in simulation and control software

for robot manufacturing [58]. The field of digital manufacturing rely on plant visualization [57]. Visualizations are put to use in various situations including control, monitoring and offline programming tasks. The typical visualization is part of a large system running on a dedicated workstation. However, the Internet has created a demand for light-weight visualizations capable of running on low-end, low-cost personal computers to form the backbone of new types of user interfaces.

6.3 Scalability

Within automation, there is currently a clear trend towards creating application software by graphically composing available software components. The issue now is to select the most promising approach to accomplish the concept of executable visualization.

Components used in industry today are mostly written in C/C++ by programmers well acquainted with the *restart-the-computer culture* which they have learned to accept; believing in the utopia that you will finally find that last error. Of course finding all faults can be done in theory, but in practice the use of an unsafe language like C/C++ implies that the engineering effort is too high. This means, for example, that even if only one out of 50 used components at some time contains a bad pointer (due to manual memory management) or an array index out of bound, that can affect data which in turn may cause the entire application to crash.

As applications get larger and more complex, and considering that components are more frequently used in safety critical application (such as hazardous chemical processes), we need to worry about safe and dependable operation. That involves several issues such as redundancy, supervisory control, error handling, etc. But more logically, to make sure those features really work, we need to ensure proper program execution. This implies that the use of unsafe languages, for other than well restricted/encapsulated local interfaces or drivers, will have to be abandoned for control systems. This is a necessary but not sufficient condition for safe operation.

For a language to be called safe, we use the definition that all possible executions are defined in terms of the language itself. This implies, for example, that it has to be abandoned to: use absolute memory addresses, create dangling pointers, index outside an array, cast-away type checking, or reference uninitialized memory. If any of this would be allowed, execution can result in something that is neither expressible as a program nor desired. We talk about core dumps, *blue screens*, and the like. A program written in a safe language can also crash, but only in a controlled way; for instance by throwing an exception to the invoking application, which

cannot be damaged by illegal memory access. Instead, measures can be taken to manage the application in an appropriate way.

When it comes to the actual control of industrial processes and manufacturing equipment, special care is needed to obtain real-time performance, and also to maintain operator interaction on the factory floor. Automatic memory management, or garbage collection, which is part of the Java program execution and a cornerstone of the scalability of Java, is often referred to as an obstacle for real-time performance. That is, however, not true. In our group it has been proved in theory, and demonstrated in practice on a real industrial robot, that well designed automatic memory management works fine and predictable even in hard real-time systems [59].

Encouraged by these results, and realizing that Internet and enterprise computing techniques are applicable even down to the field-bus level [60, 61], we have focused on operator interaction and graphical user interfaces which play a key role in programming, configuration, and operation of manufacturing equipment. As the most challenging case, we consider industrial robots and the need to handle description and presentation of geometries. We then need a technique that provides both scalable/safe operation, and visualization that scales well from powerful workstations down to dedicated devices on the factory floor.

The natural choice today is the Java language. Java has already made its way into enterprise computing, and since the same requirements show up in factory automation, Java appears to be very well suited for the task. Thus, we try to use Java for its safety, which is required to obtain scalability, and history has taught us that in the long run it is the scalable techniques that survive.

6.4 Executable visualization

We would like to put forward the notion of executable visualization as a term for software components containing a graphical description and customized code to render the description.

Four aspects of usability

A hard coupling between rendering and description provides a number of possible advantages for the user such as customized graphical descriptions, customized rendering, self-contained graphical descriptions and platform-independent execution and behaviour.

Customized graphical descriptions. A problem that has existed during a long time, but has exploded with the introduction of Internet, is the large

number of existing file formats. It is not feasible to equip each computer with readers for every file. One solution to this problem is to introduce generic file formats and have generic file viewers. This is the normal approach on the Internet today (Adobe Public Document Format for WEB publishing, HTML for browsing and VRML for 3D graphics). This solution is, however, not feasible for specialized situations needing customized functionality. The normal approach is to develop specialized software tools. We propose another way; by focusing on the data and enhancing it with custom functionality we achieve an essentially self-contained data format. It will be possible to directly export CAD geometry with enhanced functionality without worrying if the target computer has the ability to render the enhanced CAD format.

Customizable renderers. The property of customization is important as it allows executable visualizations to be customized for special tasks, with special demands on navigation, control and feedback functionality. An executable visualization has the ability to modify its renderer to incorporate customized behaviour, for instance to enhance a static graphical description with animation capabilities, to provide customized navigation capabilities for user interaction, and to incorporate external control interfaces. It is to be expected that demands on functionality will vary extremely depending on situation.

Software component packaging of graphics. A system might be heterogeneous from several different points of view: different processing power and memory capacities, different capabilities for visualizing graphics and different platforms. An executable visualization should be able to render and produce reliable results in a heterogeneous environment. Our prototype implementation achieves platform- and environment-independence by utilizing the Java and Java Beans component technology.

Template software components. Using the notion of executable visualization, it will be possible to speed up development of similar visualizations by creating a template software component containing a customized renderer and use it with all graphical descriptions. For instance, it will be possible to create a renderer providing custom functionality for rendering robot geometry and use this renderer to create executable visualizations for a whole product range of robots.

Approach to building

When building executable visualizations established technologies should be used as much as possible in order to achieve rapid development, low maintenance, and high platform independence. We therefore propose an implementation in three stages.

The first stage consists of a graphical description of the visualization, for instance CAD geometry resembling an industrial robot. It is important

that the file format of the description in this stage conforms directly to the source of the graphical descriptions. In our robot example, we want to use CAD geometry also for the robot that is to be used in the manufacturing task. The executable visualization should store the geometry in a CAD format, for instance VRML.

The second stage consists of the renderer together with customization code. The available range of renderers today goes from API-accessible renderers to pure standalone rendering applications. A problem is the customization code. Customization may potentially be provided through three sources; through changes to the graphical description file, through customization by developing a renderer based on a rendering API, or through customization of a standalone renderer. That is, there are three approaches:

The first approach involves a modifier that annotates the description file with customization code. This demands, however, that the language used to express the graphical description contains the power necessary to provide demanded functionality. Most CAD systems of today are able to export static geometry to VRML. Built into the language of VRML is the possibility of script-driven animation, something CAD systems do not utilize. The creation of a moving robot from a static robot model could, in the second layer, involve the annotation of the VRML description with scripts driving an animation. A standard renderer of VRML might then be invoked. As a counterexample, VRML is not able to express all kinds of functionality. We might want to express a customized navigation capability like semantic zooming, which is a technique for choosing the wanted detail level in a visualization. In the VRML this might only be achieved with great difficulty because the language lacks constructs for handling such functionality [62].

The second approach relies on the availability of a rendering API, preferably able to read the file format to be rendered. The advantage of having a renderer available through an API is that it provides a high-degree of freedom for customization; you are essentially free to develop your own custom renderer on top of the API. The VRML workgroup has developed a VRML rendering API running on top of Java 3D, a software package available for the Java 2 platform [63, 64, 65, 66].

The third approach uses standalone renderers with customization ability. The Cosmo player [67] (a VRML viewer) uses a link called the External Authoring Interface [68] to enable the Java language to connect to the viewer and affect the VRML model.

The third stage encapsulates the visualization into a common component technology. This enables the executable visualization to be incorporated as a component into the application, with a component tool like Microsoft Visual Basic and Java Beanbox.

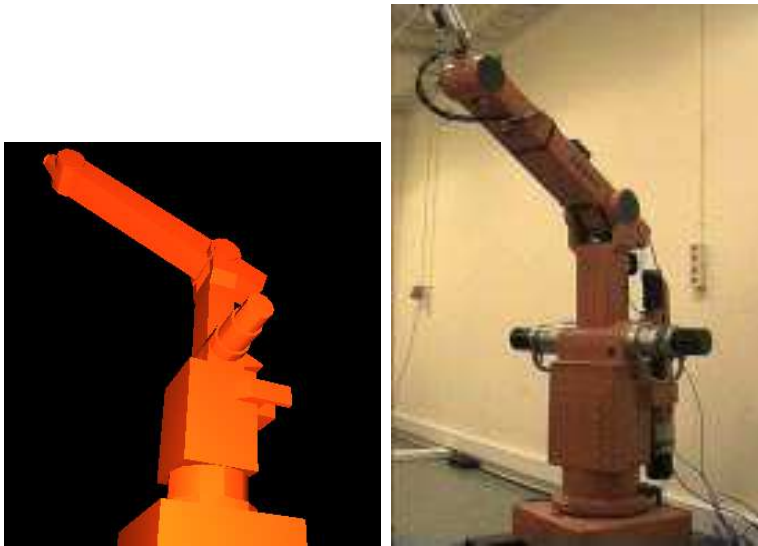


Figure 6.1 Executable visualization showing virtual ABB IRB6 robot to the left and the real robot available in our laboratory to the right.

The conclusions to be drawn from this section is that the implementation of executable visualizations as we propose them will not state a problem. However, the preferred way to do it is the second approach, using a rendering API, which provides most customization power. Some experiences from using that approach now follows.

6.5 Implementation

Component technologies such as ActiveX, JavaBeans, CORBA, COM, DCOM provides the infrastructure upon which executable visualizations may reside as natural extensions. Tools for dealing with components are quite common [69, 70]. There exists a lot of renderers (Java 3D, Cosmo, Open-Inventor, OpenGL++), converters between different file formats, and a few neutral file formats for the exchange of geometry between different systems. As mentioned, we have chosen to use VRML.

The core Java 2 platform from Sun Microsystems Inc. may be extended with a package called Java3D also available from Sun. Java3D is an API capable of rendering high-level 3D graphical descriptions onto OpenGL and DirectX platforms. Java3D is closely related to VRML. In fact, Java3D was designed to allow easy transitions from VRML to Java3D. By using

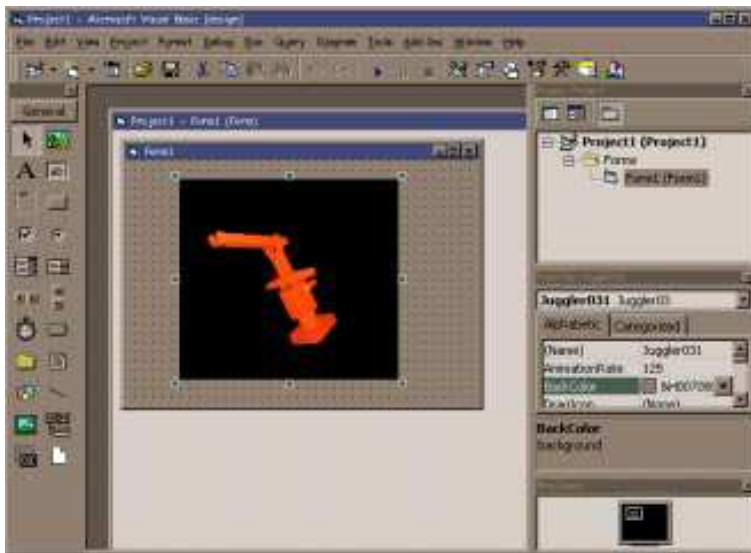


Figure 6.2 Executable visualization as an ActiveX component in Microsoft Visual Basic, using a JavaBeans-ActiveX bridge.

Java3D it is possible to encapsulate VRML graphics with a VRML renderer into a component and customize the rendering through the Java language. We have utilized this for creating virtual industrial robots available as Java Bean components or ActiveX components, executable in a number of user interface environments such as Visual Basic, Internet Explorer (HTML) and Java standalone application. Most notable is that the component due to Java runs in these different environments without modification.

An industrial robot implementation

Our prototype implementation of an executable visualization for visualizing industrial robots. The prototype is based on the Java 3D renderer enhanced with VRML. The robot is encapsulated as a JavaBean component and is able to run in an ActiveX environment through a ActiveX-JavaBeans bridge, see Figure 6.2.

The prototype expects static geometry in VRML as directly exported by one specific off-line programming system¹, but should handle VRML exported from other systems with no problem. Our example robot (ABB Irb-6) is shown in Figure 6.1. The VRML describing the robot is annotated

¹The IGrip System, <http://www.deneb.com/>

```
#VRML V1.0 ascii
Separator {
  DEF IRB-6 Separator {
    DEF IRB-6:IRB_6-0 Separator {
      MatrixTransform {
        matrix 1 0 0 0
              0 0 -1 0
              0 1 0 0
              0 0 0 1
      }
      Separator {
        Material {
          diffuseColor 1 0.38 0
        }
        Coordinate3 {
          point [
            0 0.28125 0.17,
            -0.27 0.28125 0.17,
            -0.27 0.225 0.17,
            -0.27 -0.225 0.17,
          ]
        }
      }
    }
  }
}
```

Figure 6.3 VRML geometry exported from the IGrip system. Note the named node IRB-6. That node represents a rigid body part of the robot.

with named nodes in order for the component to recognize rigid robot parts among the geometry, see Figure 6.3.

At the initial stage of our project there were no VRML loaders available for Java3D. However, several efforts are being made to create VRML loaders, the most notable being the formation of a Java3D and VRML Working group within the Web3D Consortium¹. As we saw that several implementations were on their way but were not quite ready when we needed them, we wrote a simple tool, translating VRML geometry into its Java3D equivalent. This was easily accomplished as most CAD systems only utilize a small subset of VRML when exporting geometry. The disadvantages of this solution is that we do not retain the original file format within the component and we have to recompile the component in order to change geometry.

The renderer has enhanced the original static view with animation capability. The robot is able to move its individual joints, according to data supplied at runtime. This is accomplished by "hooking" the identified rigid robot parts onto a linked structure of Java3D transform objects. Finally, the resulting objects are made into a JavaBean and transformed into

¹<http://www.web3d.org/WorkingGroups/vrml-java3d/>

an ActiveX component using the available JavaBean-ActiveX bridge from Sun Microsystems Inc.

The prototype is well suited to act as a template component to create similar visualizations for other robots, for instance an ABB Irb-2000 robot, which is also available in our laboratory. The geometry used for creating the robots may be as simple as the VRML geometry available on the ABB product page¹. A spinoff usage of the component is to visualize motion data recorded from the real Irb-6 robot.

Both the robot models and the software platform (Java 2) are freely available on the Internet.

6.6 Applications and discussion

The use of computer graphics in the personal computer market is, as mentioned, affecting the manufacturing market. Soon the use of graphics and particularly 3D graphics is not going to be a nice feature but a demand from the customer. The use of graphics in the manufacturing industry today is mostly concentrated towards large-scale visualization (digital manufacturing) and CAD system design. Since that is more or less established technology, we will now see how this scales down to dedicated end-user interfaces for use on the plant floor.

Whether or not we need a powerful online interface to a machine or robot very much depends on the manufacturing task. For instance, assembly of circuit boards is in most cases well specified from a CAD/offline model, which among other things uses a database with descriptions of the physical components to assemble. In such case, an initial calibration of the robot relative to some fixtures may be enough, and only a very simple on-line interface is needed.

In other application areas, accurate off-line modeling is much harder, for instance due to unmodeled dynamics of the manufacturing process. This is often the case within large application areas such as welding and deburring [2]. Therefore, such robots are often handled via a small user interface that the operator can carry around close to the manufacturing process, see Figure 6.4. There is of course also a tradeoff to be done between a hand-held simpler interface and a more powerful interface via, for instance, a PC connected directly to the machine. We leave that decision to the industrial development. Instead, we consider the techniques that can be applied in a flexible manner. An interesting alternative is to have a complete Windows platform even in the hand-held device. That may, however, not be the most efficient solution, but it can in any case be

¹<http://www.abb.se/robotics/product/index.html>



Figure 6.4 Robot operator/programmer at Volvo, using a hand-held terminal for on-line changes. (With permission from Volvo and ABB).

used beneath the principles we propose. As an example, let us consider an arc-welding application.

Arc-welding example

Assume manufacturing a product includes, among other things, welding two metal pieces together. Due to tolerances of the work-pieces and the difficulties to exactly predict the outcome of welding operation, there is a need for an appropriate end-user interface by which the production engineer can tune the welding operation. Such tuning may need to be different for different parts of the seam. Furthermore, there is a desire to let the operator adjust the welding in terms of the geometry of the welding seam, rather than on some less understandable voltage or wire-feed parameter. For this purpose, there is a trend towards having knowledge about the welding process stored in databases that can be accessed via the factory network.

To our knowledge, there are no really good such interface today. Instead, we base this discussion on want-lists from production engineers. Given the specifications for the user interface needed for a certain appli-

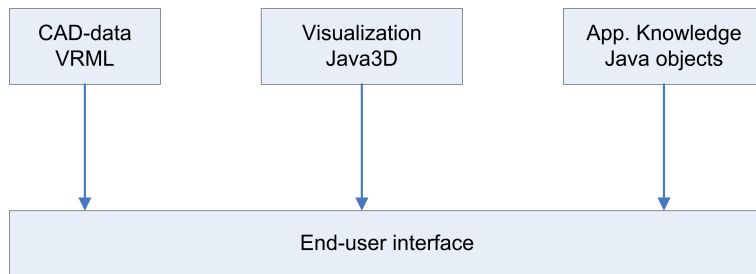


Figure 6.5 The end-user interface incorporates the geometry, the possible customized visualization and navigation, and the application know-how.

cation, one could of course implement it directly in, for example, C++ on the Win32 platform utilizing available ActiveX/DCOM components written the same way. That would, however, impose restrictions on safety and flexibility. Instead, we suggest a Java-based implementation. To further explain our approach, each of the input sources depicted in Figure 6.5 will now be reviewed.

The purpose of the CAD data input is to obtain up-to-date coordinates on which the machine/robot operations can be defined. For brevity, retrieval of calibrated coordinates back to the CAD system [2] is not treated here. A variety of existing CAD data formats exist today, but many of them are too limited for 3D description, internal proprietary, platform specific, etc. Therefore, we have chosen the VRML format; most systems can export the object geometry in VRML. The VRML format is a platform neutral ISO/IEC standard designed for the Internet.

Visualization is today mainly used for off-line programming; in on-line programming the physical equipment and work-pieces are there, so why visualize it? Reasons include:

1. To make referencing and description of coordinates during programming more user friendly.
2. To make monitoring of ongoing machine operations more understandable.
3. To tune and optimize the manufacturing process.

Items 1 and 2 are obvious but let us see what the third item could mean in arc-welding. Examples:

The welding technician may want to study the weld-seam profile along the path, both the CAD-model and the actual work-pieces, and confirm the generated welding settings from the engineering department.

In case of deviations between the model and the real world objects, there should be a way of calibrating the model, and to simulate the effect of using the welding settings (such as currents, wire-feed, path speed, and weaving amplitude). Since the seam properties changes along the seam, one can easily imagine the benefits of having a customized visualization and navigation along the track. Using Java3D with imported VRML models, in combination with a customized rendering and navigation tool appears to be very useful.

The input from the application knowledge database may concern guidelines and rules how different settings affect each other, but also specific rules and restrictions how the welding has to be performed to meet certain quality requirements [71]. Today, such functionality is data driven from tables and special data formats. This limits flexibility and complicates the implementation of end-user tools since parsing and data conversions between different formats often have to be done. Instead, we suggest the use of Java objects, for the same reasons as in enterprise computing. The advantages of obtaining not only data but also methods add a new degree of freedom in the way application know-how can be expressed. Clearly, to have embedded systems running methods dynamically loaded via the network requires a safe language.

The Java technology we use appears to be very flexible and scalable, and systems can be built more easily based on well-designed APIs and software that are freely available on the Internet. The Java objects and beans that make up a scalable application can of course also be compiled or wrapped into current Windows technology for usage in systems available today.

6.7 Conclusions

The use of computer graphics in user interfaces today poses some problems. Graphical description languages are either too low-level (OpenGL) or lack the expressiveness (VRML) needed for use in user interfaces. The diversity of description languages is another problem that causes compatibility problems between systems. This may be solved by creating customized visualization components containing both graphical description and executable code, what we call executable visualizations. The major benefit of using the notion of executable visualization is that it exploits the large body of CAD geometry to provide cheap visualizations that are easy to create, to maintain and to update by customizing the component rather than the geometry itself.

We have shown on the Java platform that it is possible to create executable 3D visualizations which animates robots exported as static geom-

etry objects from an object library while making minimal intrusion into the robot geometry description, thus providing cheap domain controlled animation to a whole product range of robots. The resulting visualization is packaged as a software component and is directly executable in a Microsoft environment, as well as in a Java environment and a browser environment. Also, the Java platform has the additional advantage of being free software. This makes it possible to freely create and distribute computer graphics in the form of Java software components.

The arc-welding example illustrates the need for this type of visualization in the industry. The need to incorporate domain specific information in user interfaces is essential for advanced control applications, and our prototype implementation shows that the proposed techniques accomplishes that in a feasible way.

Together with the portability and scalability of the Java 2 platform, our approach appears to have unique benefits, which should be of great value within the field of manufacturing.

7

A Robot Speech Interface with Multimodal Feedback

Complimentary and redundant user interface

7.1 Introduction

Industrial robot programming interfaces provide a challenging experimental context for researching integration issues on speech and graphical interfaces. Most programming issues are inherently abstract and therefore difficult to visualize and discuss, but robot programming revolves around the task of making a robot move in a desired manner. It is easy to visualize and discuss task accomplishments in terms of robot movements. At the same time robot programming is quite complex, requiring large feature-rich user interfaces to design a program, implying a high learning threshold and specialist competence. This is the kind of interface that would probably benefit the most from a multimodal approach.

This paper reports on a prototype speech user interface developed for studying multimodal user interfaces in the context of industrial robot programming. The prototype is restricted to manipulator-oriented robot programming. It tries to enhance a dialogue, or a design tool, in a larger programming tool. This approach has several advantages:

- The speech vocabulary can be quite limited because the interface is concerned with a specific task.
- A complete system decoupled from existing programming tools may be developed to allow precise experiment control.
- It is feasible to integrate the system into an existing tool in order to test it in a live environment.

The aim of the prototype is to develop a speech system for designing robot trajectories that would fit well with current CAD paradigms. The prototype could later be integrated into CAD software as a plug-in.

Further motivation lies in the fact that current available speech interfaces seem to be capable of handling small vocabularies efficiently, with performance gradually decreasing as the size of the vocabulary increases. This makes it interesting to examine the impact of small domain-specific speech interfaces on larger user interface designs, perhaps having several different domains and collecting them in user interface dialogues.

The purpose of the prototype is to provide an experimental platform for investigating the usefulness of speech in robot programming tools. The high learning threshold and complexity of available programming tools makes it important to find means to increase usability. Speech offers a promising approach.

The paper is organized as follows: speech, multimodal interfaces, and robot programming tools are briefly recapitulated. Then, the prototype is described giving the design rationale, the system architecture, the different system parts, and a description of an example dialogue design. The paper concludes with a discussion of ongoing experiments and future enhancements to the prototype.

7.2 Multimodal interfaces and robot programming tools

Speech recognition and synthesis

Speech software has two goals: trying to recognize words and sentences from voice or trying to synthesize voice from words and sentences. Most user interfaces involving speech need to both recognize spoken utterances and synthesize voice. Recognized words can be used directly for command & control, data entry, or document preparation. They can also serve as the input to natural language processing and dialogue systems. Voice synthesis provides feedback to the user. An example is the Swedish Automobile Registry service providing a telephone speech interface with recognition and synthesis allowing a user to query about a car owner knowing the car registration plate number.

A problem with speech interfaces is erroneous interpretations that must be dealt with [72]. One approach to deal with it is to use other modalities for fallback or early fault detection.

Multimodal user interfaces

A multimodal user interface makes use of several modalities in the same user interface. For instance, it is common to provide auditory feedback on operations in graphical user interfaces by playing small sounds marking important stages, such as the finish of a lengthy compilation in the Microsoft Visual C++ application. Rosenfeld gives an overview in [73].

Different modalities should complement each other in order to enhance the usability of the interface. Many graphical interfaces, including robot programming interfaces, are of the direct-manipulation type. Speech should therefore complement directmanipulation interfaces [74]. Grasso [75] lists complementary strengths and weaknesses related to directmanipulation and speech interfaces:

- Direct manipulation requires user interaction. It relies on direct engagement and simple actions.
- The graphical language used in direct manipulation interfaces demands consistent look and feel and no reference ambiguity to be usable. This makes it best suited for simple actions using visible and limited references.
- Speech interface is a passive form of communication. The medium allows for describing and executing complex actions using invisible and multiple references. It does not require use of eyes and hands making it suitable for hand-eye free operations.

Put in other words: speech might be used to avoid situations where you know exactly what you want to do but do not have a clue as where to find it in the graphical user interface. It may also help to avoid situations when you are able to describe an operation but do not know how it is expressed in the user interface.

Industrial robot programming interfaces

Essentially all robot programming boils down to the question of how to place a known point on the robot at a wanted position and orientation in space at a certain point in time.

For industrial robot arms, the known point is often referred to as the tool center point (TCP), which is the point where tools are attached to the robot. For instance, a robot arm might hold an arc-welding tool to join work pieces together through welding. Most robot programming tasks deal with the specification of paths for such trajectories [76].

Below is discussed how modeling of trajectories is performed in three different tool categories for programming industrial robots.

Teach pendant A single robot operated by a person on the factory floor is normally programmed using a handheld terminal. The terminal is a quite versatile device. For instance, the ABB handheld terminal offers full programmability of the robot. The terminal has a joystick for manual control of the robot. Function buttons or pull-down menus in the terminal window give access to other features. Program editing is performed in a syntax-based editor using the same interface as for manual operation, i.e. all instructions and attributes are selected in menus. Special application support can be defined in a way uniform to the standard interface.

Trajectories are designed by either jogging the robot to desired positions and record them or by programming the robot in a programming language. For ABB robots the programming language used is called RAPID [77].

Off-line programming and simulation tools In engineering environments, programming is typically performed using an off-line programming tool. An example is the Envision off-line programming and simulation tool available from Delmia. These tools usually contain: An integrated development environment. A simulation environment for running robot programs. A virtual world for visualizing running simulations and being used as a direct manipulation interface for specifying trajectories.

Trajectories are designed by programming them in a domain-specific language or by directly specifying points along the trajectory. The simulation environment provides extensive error checking capabilities.

CAD and task level programming tools Task level programming tools typically auto-generate robot programs given CAD data and a specific task, for instance to weld ship sections. The software works by importing CAD data and automatically calculate necessary weld trajectories, assign weld tasks to robots and generate programs for these robots. These tools are typically used for programming large-scale manufacturing systems.

7.3 Prototype

Two observations can be made concerning the user interfaces in the above programming environments: The typical task performed by all IDEs (Integrated Development Environment) is to model task specific robot trajectories, which is done with more or less automation, depending on tool category. The user interface consists of a visualization and a programming part, see Table 7.1.

The prototype presented here is a user interface where speech has been chosen to be the primary interaction modality but is used in the presence

Table 7.1 Visualization and programming in different categories of robot programming tools.

<i>IDE</i>	<i>Visualization</i>	<i>Programming</i>
Teach pendant	Real env.	Jogging & lang.
Off-line tool	Virtual env.	Lang. & sim.
Task-level tool	Virtual env.	CAD data

of several feedback modalities. Available feedback modalities are text, speech synthesis and 3D graphics.

The prototype system utilizes the speech recognition available in the Microsoft Speech API 5.1 software development kit. The SAPI can work in two modes: command mode recognizing limited vocabularies and dictation mode recognizing a large set of words and using statistical word phrase corrections. The prototype uses the command mode. It is thus able to recognize isolated words or short phrases [78].

The system architecture uses several applications (see Figures 7.1, 7.2, 7.3, 7.4): The Automated Speech Recognition application, which uses SAPI 5.1 to recognize a limited domain of spoken user commands. Visual feedback is provided in the Voice Panel window with available voice commands. The Action Logic application, which controls the user interface system dataflow and is the heart of the prototype. The Text-to-Speech application synthesizing user voice feedback. The XEmacs application acting as a database of RAPID commands and also allowing keyboard editing of RAPID programs. The 3D Robot application providing a visualization of the robot equipment.

A decision was made to not use any existing CAD programming system in the prototype. The reasons were twofold: extending an existing system would limit the interaction into what the system allowed, making it difficult to easily adjust parameters like the appearance of the 3D world and the behavior of the editor. The second reason is that by not including a commercial programming system it is possible to release this prototype into the open source community as a complete system.

System architecture

The prototype system architecture follows a traditional client-server approach. The action logic application acts as a server with all other applications acting as clients. Interprocess communication is performed using Microsoft Win32 named pipes and sockets.

The system dataflow is centered around the speech applications since it is the primary modality of the system. Basically information flows from



Figure 7.1 SAPI 5.1 speech interface application front end with a list of available command words.

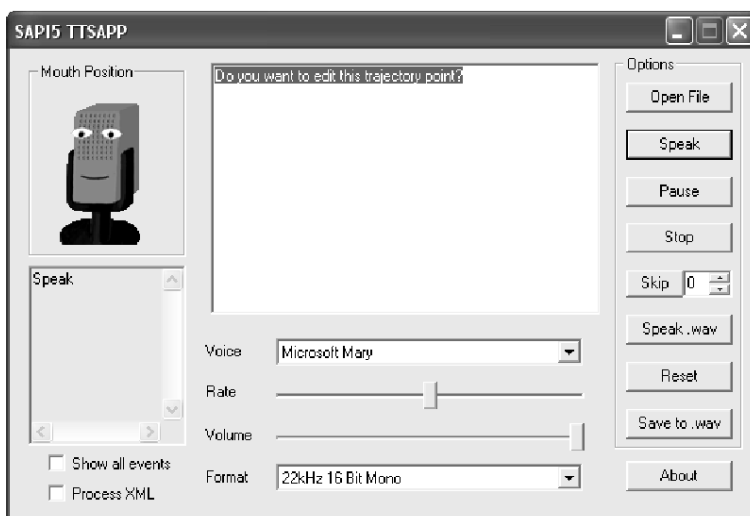


Figure 7.2 The SAPI 5.1 sample TTS application modified for use by the prototype system.

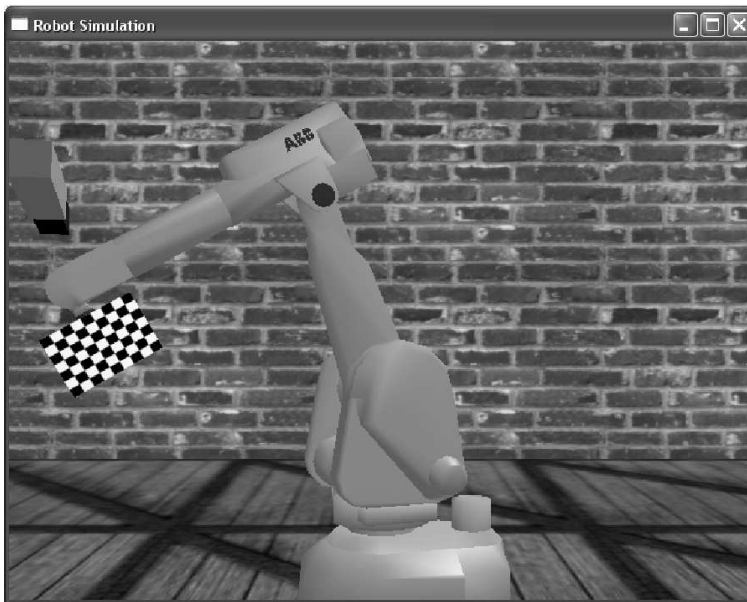


Figure 7.3 Virtual ABB IRB 2000 industrial robot arm with 6 degrees of freedom (developed in cooperation with Tomas Olsson, Dept. of Automatic Control, Lund University, email: tomas.olsson@control.lth.se).

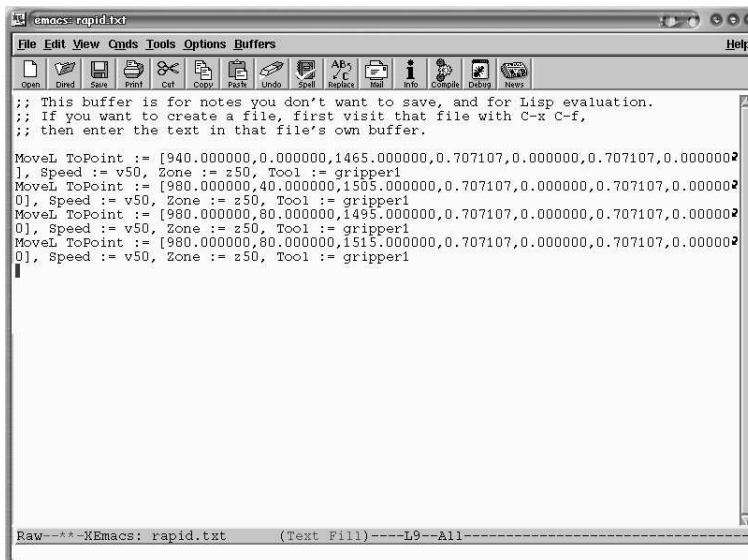


Figure 7.4 XEmacs is used as trajectory editor and database.

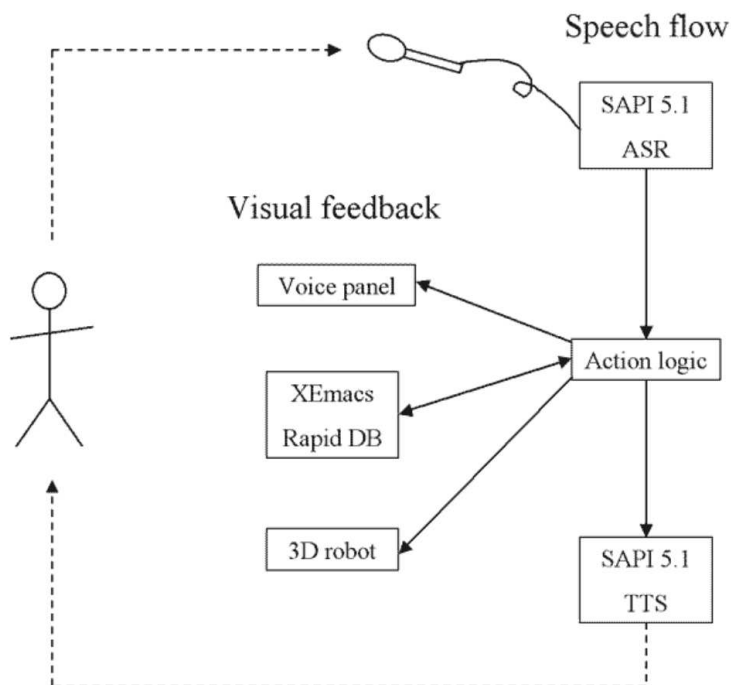


Figure 7.5 Prototype system dataflow.

the speech TTS to speech synthesis application through the action logic application. The action logic application then interacts with the other applications (XEmacs, 3D robot) in order to update the state and different views supported in the interface (Figure 7.5).

Prototype applications

Action Logic The action logic application is the heart of the system. All information goes through this application. The logic controlling the interface is hidden here.

The basic work flow of the application is:

1. Receive spoken commands from the speech recognition application.
2. Interpret the commands and act accordingly: Send Lisp editing commands to the XEmacs editor that is storing the trajectory as a sequence of RAPID MoveL (Move Linear) commands. Read trajectory stored in XEmacs and send it to the 3D application for execution

and visualization. Send feedback to be spoken to the speech synthesis application.

Microsoft SAPI 5.1 speech recognition and synthesis The speech recognition and synthesis applications are based on the Microsoft Speech API version 5.1. Each application is built by utilizing an example application delivered together with the SDK and modifying it for our purposes. The example applications used for the prototype are CoffeeS0 and TTSAApp.

The modifications necessary were quite small. They included: Adding communication capabilities to the applications so that they could send and receive information from the action logic application. This was done by adding a new communication thread to the application. Modifying the application window message handler to issue and receive speech messages from the new communication code. Changing the user interface to show our domain-specific vocabulary. And finally tune the speech recognition application to our vocabulary. This was done by rewriting the default XML grammar read into the speech recognition application upon initialization.

XEmacs RAPID trajectory editing and database XEmacs is utilized as a combined database, editing and text visualization tool. The trajectory being edited is stored in an XEmacs buffer in the form of a sequence of RAPID MoveL commands:

```
MoveL ToPoint := [940,0,1465,0.707,0,0.707,0],  
Speed := v50, Zone := z50, Tool := gripper1  
MoveL ToPoint := [980,80,1495,0.707,0,0.707,0],  
Speed := v50, Zone := z50, Tool := gripper1
```

The trajectory code is visualized in text form in the XEmacs buffer window. It may be edited using normal XEmacs commands. Thus the interface, even if developed with speech in focus, allows alternate interaction forms.

The interaction between XEmacs and the action logic application is done using LISP, see Table 7.2. The action logic application phrases database insertion/removal/modification commands of trajectory parts as buffer editing commands. These are executed as batch jobs on the XEmacs editor using the gnuserv and gnuclient package.

Virtual environment The prototype needed a replacement for the 3D visualization usually shipped with robot programming applications to be realistic. A small 3D viewer previously developed was taken and enhanced with interpretation and simulation capabilities for a small subset of the RAPID language.

Table 7.2 Sample LISP editing command sent to the Emacs RAPID database in response to spoken commands.

<i>Spoken command</i>	<i>Emacs LISP</i>
Add point	(kill-new "MoveL..."), (yank)
Remove point	(kill-entire-line)
Move forward	(forward-line 1)
Move backward	(forward-line -1)

Table 7.3 Vocabulary used in the prototype.

<i>Spoken commands</i>	<i>Purpose</i>
Forward, backward, left, right, up, down	Jog robot
Play, stop, step forward, step backward, faster, slower	Play trajectory
Mark, add point, move point, erase point	Edit trajectory
Yes, no	User response
Undo	Undo

The tool is capable of acting as a player of trajectories stored in the XEmacs database. Player commands (play, reverse, stop, pause) is controlled from the action logic application.

Dialogue design

A preliminary experiment based on Wizard-of-Oz data obtained from the authors has been implemented.

The basic idea of this interface is to view trajectory modeling as editing a movie. It is possible to play the trajectory on the 3D visualizer, insert new trajectory segments at the current point on the trajectory, remove trajectory segments, and moving along the trajectory backward and forward using different speeds.

All editing is controlled using spoken commands, see Table 7.3. The user gets feedback in the form of a synthesized voice repeating the last issued command, seeing the trajectory in text form in the XEmacs buffer window and seeing the trajectory being executed in the 3D window. The command is always repeated by a synthesized voice in order to detect erroneous interpretations immediately. At some points (for critical operations like removal of trajectory segments), the interface asks the user if he/she wants to complete the operation.

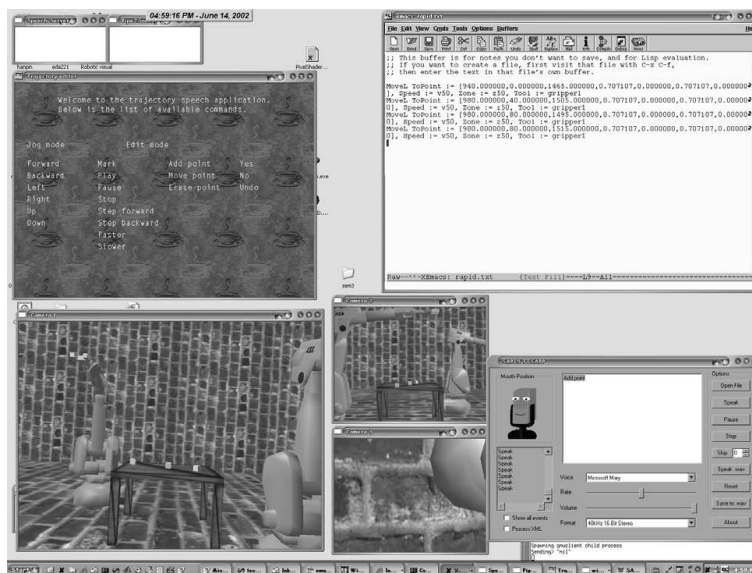


Figure 7.6 The prototype system user interface consists of four windows; 1. The voice panel containing lists of available voice commands. 2. The XEmacs editor containing the RAPID program statements. 3. The 3D visualization showing the current state of the hardware. 4. The TTS application showing the spoken text.

7.4 Ongoing experiments and future work

The prototype will be used to explore the design space of speech interfaces with multimodal feedback. Below follows a few issues that would be interesting to gather data on:

- Varying the degree of voice feedback, as well as the type of information conveyed.
- Varying between different kinds of visual feedback.
- Varying the command vocabulary and interface functionality. For instance by allowing some task level abstractions in movement specifications, i.e. move to object, grab object.

For the future, there is a list of wanted extensions:

- Allow multiple domains in the speech recognition application, with the option of choosing which one to be applied from the action logic application. This feature could be used to test speech interfaces with state.

- Allow the entire experiment interface configuration to be specified in XML. Remove all hacking necessary to tune the interface. This would also speed up development since it would be easy to switch between different configurations.

7.5 Conclusion

We have developed a speech interface to edit robot trajectories. An architecture based on reusable application modules was proposed and implemented.

The work is aimed at studying feasibility and usefulness of adding a speech component to existing software for programming robots. Initial feedback from users of the interface are encouraging. The users, including the authors, almost immediately wanted to raise the abstraction level of the interface by referring to objects in the surrounding virtual environment. This suggests that a future interface enhancement in such direction could be fruitful.

8

Human-Robot Interaction using Advances Devices

User interface for configuration using digital paper

8.1 Introduction

The success of using robots with flexible manufacturing systems especially designed for small and medium enterprises (SME) depends on the human-machine interfaces (HMI) and on the operator skills. In fact, although many of these manufacturing systems are semi-autonomous, requiring only minor parameterization to work, many other systems working in SMEs require heavy parameterization and reconfiguration to adapt to the type of production that changes drastically with time and product models. Another difficulty is the average skill of the available operators, who usually have difficulty adapting to robotic and/or computer-controlled, flexible manufacturing systems. This reality configures a scenario in which flexible automation, and robotics in particular, play a special and unique role requiring manufacturing cells to be easily used by regular non-skilled operators, and easier to program, control and monitor. One way to this end is the exploitation of the computer input-output devices to operate with industrial robotic equipment. With this approach, developers can benefit from the availability, and functionality of these devices, and from the powerful programming packages available for the most common desktop and embedded platforms. On the other hand, users could benefit from the operational gains obtained by having the normal tasks performed using common devices, and also from the reduction in prices that can arise from using consumer products [79].

Industrial manufacturing systems would benefit greatly from improved interaction devices for human-machine interface even if the technology is

not so advanced. Gains in autonomy, efficiency, and agility would be evident. The modern world requires better products at lower prices, requiring even more efficient manufacturing plants because the focus is on achieving better quality products, using faster and cheaper procedures. This means having systems that require less operator intervention to work normally, better human-machine interfaces, and cooperation between humans and machines sharing the same workspace as real coworkers [79].

Also, the robot and robotic cell programming task would benefit very much from improved and easy-to-use interaction devices. This means that availability of SDKs and programming libraries supported under common programming environments is necessary.

This paper explores the utilization of programmable digital pens [80, 81] for the task of programming industrial robot manipulators. The objective is to demonstrate the usefulness and potential of these devices, defining a suitable platform that could be used to implement user services to support new and advanced features.

8.2 The Anoto Digital Pen

A digital pen is a device that stores the user hand writing in a digital format, providing a mechanism for the user to access the stored data from a computer. The concept developed by the Anoto [80] is very innovative and suitable for robotic applications. The technology developed by Anoto, entailing interpretation and transmission of handwritten text and images, is based on a special digital pen and a paper printed with a pattern that is almost invisible to the eye (Figure 8.1). The digital pen uses ink and handles just like a normal ballpoint pen, but it also contains a digital camera, an advanced image processing system and a communication unit, for example for wireless Bluetooth connection to a mobile phone.

The paper consists of an ordinary paper printed with a dot pattern that is either pre-printed or printed on a laser printer. The displacement of the dots, 0.1 millimeters in size, from the relative position enables them to be programmed to tell the pen the exact location on the page – or the whole pad of papers – one is writing on.

Briefly, a device like this allows the user to extract the following information (Figure 8.2):

1. Absolute position of the pen on the paper page.
2. The sequence of pen movements as a map of coordinates.
3. Each coordinate has information about the position, the time stamp (both relative to the start of the stroke and current time when it is

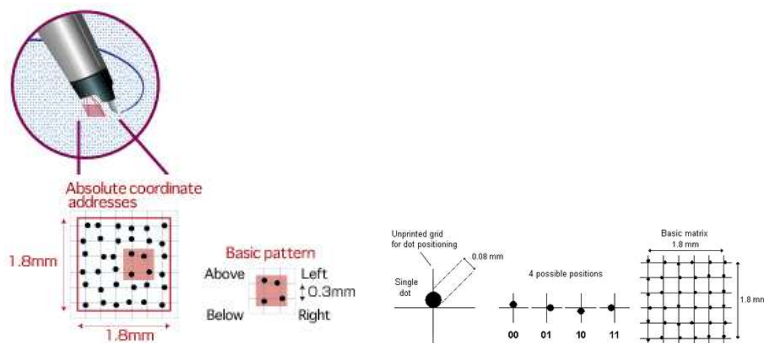


Figure 8.1 Dot pattern used by the digital pen [80].

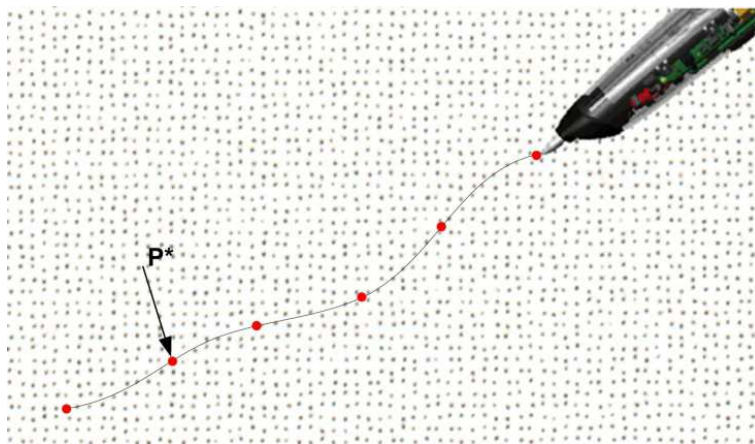


Figure 8.2 Sample line stroke: P* contains position, time, status and stroke properties information.

written), and the line color and width.

By registering the pen's movement across the paper, and also the pressure, the writing is interpreted and digitalized. Hence the technology is not based on characters having to be written in a special way, in contrast to various other applications such as hand-held computers. Even drawn sketches can be interpreted and transferred. Since the pen's movements are stored as a series of map coordinates and the paper defines where on the paper one is writing, it is possible to go back and complete previous notes in a pad. The technology is capable of interpreting this correctly and putting it in the right context when transmitting it to the digital media. The pattern enables different parts of the paper to be assigned different

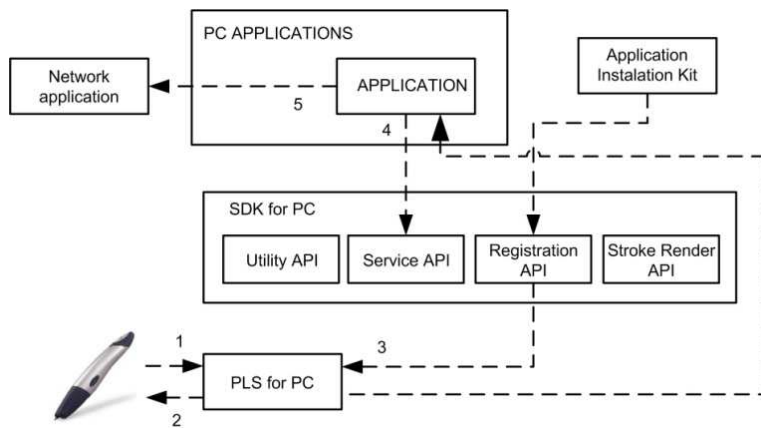


Figure 8.3 The PC architecture used by the digital pen [80].

functions such as predefining various parts of a form.

Consequently, if the user technical drawing is a robot path over a CAD drawing, for example, then the extracted sequence of positions and time stamps, etc., is a complete definition of the motion specified by the user. Selecting a proper set of reference points/orientations this information can be used to allow users to specify a sequence of robot actions just by drawing them on a paper. If a few rules are observed the information extracted from the pen can be used to generate robot programs, i.e., the user can program a robot just by drawing the robot actions on a paper, like it's common for any engineer. Anoto developed a software platform (Figure 8.3) to interface their pens with personal computers, pocket PC's, mobile phones, etc., and released the necessary APIs to allow users to fully explore the device features [82, 83].

This paper explores the digital pen as a robot input device, showing 3 different approaches/implementations designed to program robot manipulators. The first implementation presented explores the possibility to specify robot motions on CAD drawings, and using specially design computer applications to include the robot paths with the CAD package, generate the robot program code, download it to the robot controller and run it remotely. The second approach moves toward creating an industrial test case instructing cutting operations in a foundry application using the digital pen based on the Anoto technology. The third implementation deals with the generation of CAD data and robot programs from technical hand drawings. Geometric shapes and their measures are taken from a simple sketch. The user is able to define robot programs without dealing with complex interfaces. This implementation uses the Anoto platform

and uses the generated PGC (Pattern Generated Coordinates) files as input, which contains the data from the pen and is obtained when the user synchronizes the pen with the PC infrastructure (just by putting the pen in the docking station or requesting synchronization by Bluetooth).

Using the Anoto APIs an application was designed [84] to perform the following services:

1. Extract the data from the PGC file.
2. Import the user strokes to AUTOCAD [85] using the specified file and layer. The strokes are printed in the screen using also the user selected color.
3. Generate the robot code based on the received information, using two different operating modes: Free mode, where the strokes are sent to the robot as they come from the pen and the robot is commanded to move in line between points, and Corrected Mode, where the type of stroke is identified and mapped to known robot types of motions (linear, arc, circular, etc.).

The demonstration uses a welding robot (ABB IRB1400 robot equipped the S4 robot controller [86]), connected to a local area network and controlled from a laptop using software based on Remote Procedure Calls (RPC). The working table is modeled in AUTOCAD and the user is able to:

1. Draw robot paths in 2D using a CAD printout of the working table and the Anoto digital pen.
2. The obtained robot paths are transferred to AUTOCAD and identified using the procedure explained above.
3. An application is used to generate the robot code necessary to run the programmed paths, using a procedure as explained above.
4. Download the generated robot code to the robot controller and the execution commanded from the PC.

This example, developed at the Mechanical Engineering Department of the University of Coimbra shows fully the usefulness of this type of devices for robotic programming applications [87].

8.3 Development Towards Foundry Applications

Due to the bad working conditions, cutting and deburring in foundry industries is an important robot application, and this is one of the key

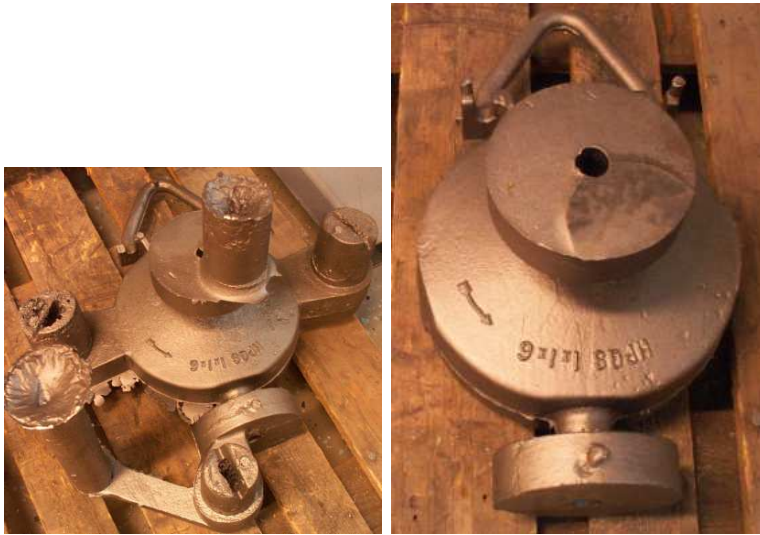


Figure 8.4 Example foundry workpieces. Left: before cutting. Right: after cutting.

demonstrators for new easy-to-use robots. To validate the usefulness of the digital pen and paper approach for robot programming and interaction in that area, initial experiments have been carried out using the work pieces from one of our end users (Norton Casts in the UK). The application includes material removal by cutting the input metal reservoirs, which are needed for the casting process but not part of the product (compare left and right picture in Figure 8.4). A second stage of the process is deburring, which we also think can be simplified by using the Anoto technology, but in this paper we focus on specification and/or programming of the cutting.

Since the foundry workpieces are produced in very small batches (1-10), it is desirable to find a quick and easy way of specifying the task such that it can be carried out by the automation equipment, including the robot, the fixture, and the cutting tool (oxyfuel burner in our case). An initial task specification experiment is shown in Figure 8.5. The experienced simplicity of system development and data import, in combination with the natural and simple way the system can be used, clearly indicate the potential and applicability of the suggested techniques.

The data to the lower left in Figure 8.5, as obtained from the Anoto interfaces, is then mapped to the coordinates used in the robot program. This is similar to the other applications in this paper.

Taking a closer look at the system and further developments, Figure



Figure 8.5 Foundry workpiece experiment. In upper picture the pen with the CAD model of the foundry workpiece printed on the paper enabling the Anoto technology and the pen strokes specify the cuts to be made. In the lower part, the imported pen strokes (left) and the surface mapping on workpiece 3D image.

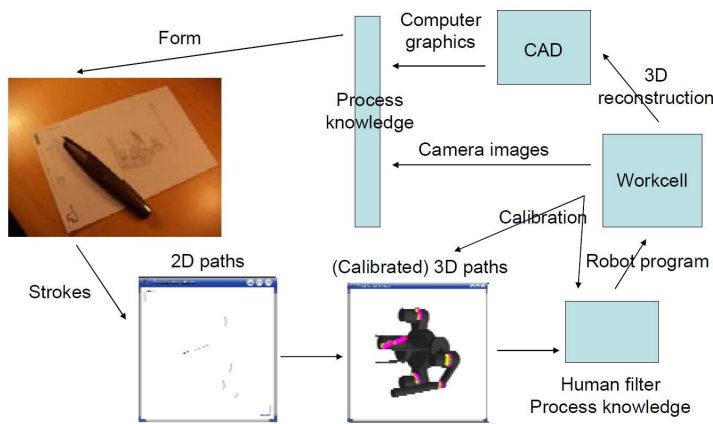


Figure 8.6 Approach for further development. Colored boxes are part of ongoing work, whereas the non-colored boxes are part of the initial experiments.

8.6 shows an example of how the data flow could look like in the prototype system. The layout and contents of the form enabling the Anoto technology need to be defined to allow efficient definition of the task. In order to do that one or several images or poses of the workpiece is necessary to print on the paper enabling the Anoto technology, however, in the foundry industry CAD models featuring metal reservoirs are very rarely produced, indicating that a method is necessary for either reconstructing geometry or using camera images of the workpiece directly printed on paper. The filled in form needs post processing for interpreting the obtained data (pen strokes), paths or symbols.

Based on these findings the proposed approach for further development

includes:

1. Obtain metal reservoir workpiece geometry and add to CAD model.
2. Assemble and print form for defining cut task.
3. Interpret pen strokes and identify cut paths on the workpiece.
4. Generate the robot program

Note that the end user simply works with pen and paper in combination with images and the real equipment; no programming skill required.

8.4 Conclusion

In this paper the authors explore the utilization of digital pens for the task of programming industrial robot manipulators. Three different implementations were presented to assess the digital pen features using different robotic setups. The results clearly show that the digital pen based on Anoto technology is very useful and powerful, being an interesting solution for certain advanced applications involving the task of programming robotic installation just by making technical drawings on a sheet of paper. The next steps will be to adopt a software infrastructure and develop the necessary services to allow system integrators to consider this type of device an advanced user-friendly user programming method.

Part IV

Semantic Robot Interfaces

9

Toward Ontologies and Services for Assisting Industrial Robot Setup and Instruction

Automatic generation of user interfaces from product and process knowledge

9.1 Introduction

Traditional robotics supports long-batch production and requires skilled personnel to handle setup and instruction. On the contrary, new robot markets often involve shorter series and small and medium enterprises. This means that the shift of products is faster and the change-overs often need to be carried out by non-experts. This sets new challenges for the robot user interfaces to be more intuitive and user friendly in order to reduce number of errors and cost/time [88]. Such challenges outline the need for assistive systems within the robot cell to make the operator less dependent on expert knowledge and turn complex tasks feasible. Examples that could benefit from assistance include calibration of tools, fixtures, and workpieces; usage of CAD/CAPP/CAM software such as task planners; configuration of process-specific software packages, such as the ABB palletizing PowerPac.

In this paper, we present an assistive infrastructure for robot setup and instruction that attempts to address these challenges. We introduce the ongoing development of a system based on semantic web technologies that automatically generates multimodal dialogue interaction. We also

describe a prototype that currently generates two modalities, digital paper and spoken dialogue.

The purpose of an assistive system is to enhance the usability and usefulness of the robot and its connected resources (sensors, CAD/CAPP/CAM systems) through:

- The use of semantic standards in information exchange, such as RDF/S, OWL, SPARQL, and SWRL;
- Production documents such as product and process data including a semantic layer;
- The definition of compatible semantic layers so that they can be used across the relevant tasks, such as aiding cell calibration and robot instruction.
- Increased automation of tasks using the semantic layer, such as finding calibration sequences to make sure nothing is forgotten.

The roadmap we follow to implement the assistive infrastructure is based on the use of an ontological network to encapsulate knowledge about the product data and manufacturing processes. It requires the derivation of ontology concepts that will serve as the main data source to generate – or refer to – the complete specifications and the operating instructions used to automate information management necessary for task planning and execution.

9.2 Multimodal Form-based Dialogue

From the user viewpoint, the operation starts with an initial selection command from the operator to tell the machine which work piece to produce and possibly from positions and equipment data sensed by devices on the floor.

After the initial selection, the system extracts data from the ontology that enables the operator to configure the task and the product, and to prepare the task execution. The configuration step uses a multimodal interface that lets the operator fill in the different options. It ends with the monitoring and execution of the configured task.

The process flow uses conversion tools such as transformation rules, inference rules, and the JastAdd compiler [89] to select and convert portions of the ontology. This results in process and product data divided into configurable and nonconfigurable parts (Figure 9.1). The extracted data are first formatted as an XML document corresponding to a production sketch that we call the XML appconf.

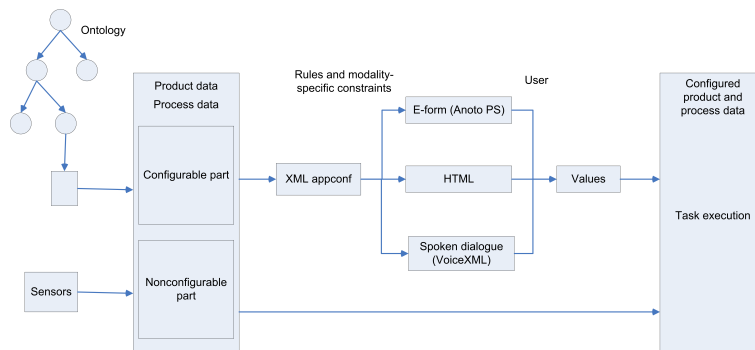


Figure 9.1 The process workflow.

From this sketch, the system automatically generates user interfaces with multiple input modalities for all the parameters. A first demonstration prototype will be available in Spring 2008.

9.3 Ontology Modeling

In computer science, ontologies correspond to hierarchical models to represent concepts, objects, and their relationships. They enable systems to [90]:

- Encode and interpret data using a rich hierarchical and relational structure.
- Extract data and integrate them into applications.
- Share data with a common format.

As ontology modeling language, we have chosen the web ontology language [91] and the Protégé toolkit [92] as a data entry and validation tool. Both are well established standards in their domain with a large developer's community. We are currently using them to build the ontology of a specific domain shown in Figure 9.2 that serves in the demonstration prototype. This ontology acts as an advanced data repository for the product configuration and the production operations. In the future, we will populate ontologies from manual modeling, specification databases, and 3D models. In addition to what we develop within SMERobot [93], the data model we will use could also possibly benefit from work being carried out at LTH for the SIARAS project [94] on production ontologies.

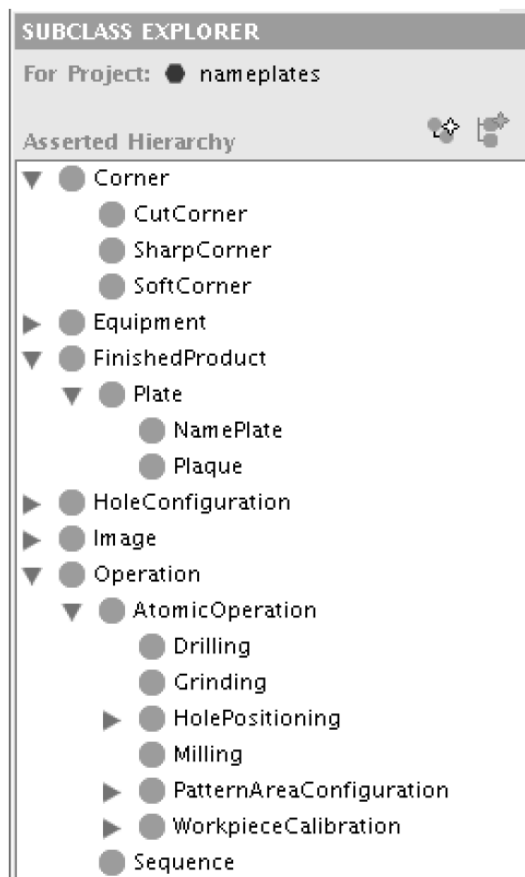


Figure 9.2 An excerpt of the ontology detailing the operation hierarchy.

The conversion pipeline shown in Figure 9.1 uses W3C recommendations associated with the semantic web such as XSLT or SWRL. The choice of these tools needs some clarification. We first summarize the concepts that are around OWL and then explain the conversion principles.

Resource Description Framework – RDF

OWL is based on the resource description framework, RDF [95]. RDF models statements as triples in the form of a subject, a predicate (a verb), and an object. As an example, the statement *the milling process starts with a calibration* can be split into a subject, *the milling process*, a predicate, *starts with*, and an object, *a calibration*. Such triples are also named, respectively, the resource, the property, and the value. RDF is restricted to

binary predicates.

This framework can use two encodings. The first one, called Notation 3, consists of sequences of textual triples and the second one adopts a XML syntax. The subject – the resource – must be an URI. The predicate or property, which is also a resource, is an URI too. The object or value can be a resource or a literal. To represent the example above, we use the lrc namespace – Lund Robotics Core – and URI <http://cs.lth.se/lrc-ontologies/1.0/>. This URI is still nonexistent when this article is being written. Using Notation 3, we can represent the example above as:

```
@prefix lrc: <http://cs.lth.se/lrc/ontologies/1.0/>.
<#milling_process> lrc:starts_with "calibration";
```

And in XML syntax as:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:lrc="http://cs.lth.se/lrc/ontologies/1.0/">
  <rdf:Description rdf:about="#milling_process">
    <lrc:starts_with>calibration</lrc:starts_with>
  </rdf:Description>
</rdf:RDF>
```

More generally, triples (subject, predicate, object) can be encoded in Prolog or Datalog as *predicate(subject, object)* that is, with the sentence above as:

```
starts_with(milling_process, calibration)
```

RDF Schema – RDFS

RDF schema, RDFS, is built on RDF and defines two predicates that enable the programmer to build an ontology: *rdfs:Class* and *rdfs:subClassOf* [96]. The *rdfs:Class* element allows to declare a RDF resource as a class and the *rdfs:subClassOf* element allows to declare subclasses of a class and build a hierarchy. When a resource has been declared as a class, we can use *rdf:type* to create individual members of this class. In addition, RDFS comprises similar predicates to build a property hierarchy.

In addition, RDFS has constructs to type the subject and the object of the triples. It corresponds to the domain and range of a function, and in the case of RDFS applies to properties. RDFS uses the constructs *rdfs:domain* for the subjects and *rdfs:range* for the objects to restrict the values of the two arguments of a property.

OWL

Although the combination of RDF and RDFS forms an ontology language, it lacks some features to build large, realistic ontologies. They include cardinality restrictions, Boolean operations on classes, etc. In addition, RDF and RDFS are not well coupled to logic and reasoning tools.

The web ontology language, OWL, is an extension of RDF and RDFS that attempts to complement them with better logic foundations and a support for practical reasoning [91]. It comes with three flavors of increasing expressivity – light, description logic, and full – that are upward compatible. Only the two first ones are guaranteed to be tractable in practice.

Important constructs of OWL include the *owl:Class*, which is derived from the *rdfs:Class*, two properties, *owl:ObjectProperty* and *owl:DatatypeProperty*, that relate objects to respectively another object or a data type value like a string, an integer, a float, etc., *owl:Restriction* that enables the programmer to use existential and universal quantifiers and cardinality.

9.4 Prototype Setup

Nameplate Manufacturing

As described in [97], the ontology programming approach uses automatically generated forms to select and configure both the task and the product. The prototype selected to demonstrate the concepts associated with our approach corresponds to the manufacturing of wood nameplates.


Figure 9.3 shows a configuration form where the left column configures the shape and looks of the plate. The right column configures the process for manufacturing the plate. In this example it is possible to skip steps, execute in a stepwise manner, and choose data acquisition methods for steps involved. The left column can be filled out at an earlier date while the right column is filled out close to task execution time. The upper right barcode identifies the process and is possibly unique to individualize the sheet.

Nameplate Manufacturing Ontology

We have built a prototype ontology to encode the process templates and we are developing well-defined conceptual interfaces toward work cells (equipment, capabilities, communication) and process data, assisting construction of process templates, and assisting (manual/automatic) work cell reconfiguration.

Wood sign process – Configuration Sheet


Start




Stop

Shape

Sharp comers




Soft comers



Comer diameter [mm]:

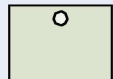
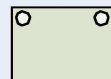
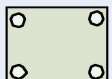
Cut comers



Comer height [mm]:



Holes configuration

1 2 4

Pattern and text

Pattern 1 Pattern 2

Text

Workpiece calibration

None (preprogrammed)

Manual (3 points)

Automatic (touch sensor)

Milling

Skip

Execute immediately

Hole configuration

Skip

Automatic

Lead-through

Gesture

Drilling

Skip

Execute immediately

Pattern area configuration

Skip

Automatic

Lead-through

Gesture

Grinding

Skip

Execute immediately

Figure 9.3 Prototype form for manufacturing wood nameplates.

The ontology is restricted to the prototype domain and Figure 9.2 shows an excerpt of it. It describes the finished products, its components, together with possible features as well as the operations involved in the manufacturing process of the product.

Intermediate Appconf Representation

Given a product to manufacture, the conversion process extracts an intermediate, flat representation from the ontology. This representation is designed to be modality-independent, which makes it easier to build user views (forms, speech, gestures). We call it the application configuration – appconf.

As an example, in the application prototype we are building, the sheet requires the operator to supply data, such as the corner shape, the hole configuration, and the pattern and text (person name for instance). All these items are shown as input areas on the sheet in Figure 9.3. The intermediate appconf representation has corresponding elements representing all these configurable items, for instance the corner shape.

We give an idea of how to represent the corner shape options in the XML code snippet below. This code replicates the possible options, sharp, soft, or cut corners, with the images to display in the e-form using the *img* element and the messages to utter using the *snd* element in the case of a spoken dialogue.

```
<shape>
  <one-of>
    <option>
      <name>sharp corners</name>
      <command code="sharp.cd"/>
      
      <snd src="sharp.wav"/>
    </option>
    <option>
      <name>soft corners</name>
      <command code="soft.cd"/>
      
      <param name="diameter" unit="mm"/>
    </option>
    <option>
      <name>cut corners</name>
      <command code=".cd"/>
      
      <param name="height" unit="mm"/>
    </option>
  </one-of>
</shape>
```

Using this configuration sketch, presentation rules, and modality specific constraints, the conversion process produces displayable forms or

spoken dialogues so that the operator can supply the missing parameters. Once the operator has filled in the data, the corresponding XML fragment is:

```
<shape>
  <name>sharp corners</name>
  <command code="sharp.cd"/>
  
  <snd src="sharp.wav"/>
</shape>
```

Methods and Languages to Extract Information from Ontologies

The conversion pipeline extracts and infers information from the ontology and generates the user input modalities. The appconf configuration sketch is an XML intermediate document between the ontology and the user interfaces. Unlike the sketch, the ontology is a structured and hierarchical representation, where features are shared and inherited across a variety of pieces and processes. This means that extraction is not trivial because the representation languages involve three complex and intricate layers: RDF, RDFS, and OWL.

Such a setting requires specific query languages and techniques. In addition to the ontology management, we need to process other XML documents in the processing chain such as the appconf sheet to convert them into forms or dialogue programs. We review here techniques and their application in the management of information along the conversion chain. They include accessing XML nodes, querying RDF triples, and reasoning about the ontology knowledge. Most difficulties come from the apparent masses of “solutions”. Wikipedia lists not less than 11 different RDF query languages and ten OWL reasoners! We focus here on what have become the (likely) standards in their respective ecosystems.

XSLT The simplest way to access and transform XML documents is to use the combination of XPath and the extensible stylesheet language transformations, XSLT [98]. XPath enables programmers to express a path and access nodes in an XML tree, while XSLT defines conversion rules to apply to the nodes. A typical application of XSLT is the transformation of XML documents into XHTML files destined to be read by web browsers.

Provided that the amount of paraphrasing (syntactic variation) is limited, XSLT XPath is fairly usable to run the conversions. From studies we have done, this is the case for the conversion of the appconf sketch to the user modalities. We are completing the implementation and integrating it in the prototype.

However, this is not the case for ontologies. They are built on OWL, which is built with RDF triples, which allows reformulating similar structures using different constructs. Querying ontologies require either query languages or reasoning rules. For a justification, see [99], pp. 100-102.

SPARQL SPARQL [100] is a RDF query language. It enables the programmer to extract RDF triples using the SELECT keyword where the variables are denoted with a questionmark prefix using a set of conditions defined by the WHERE keyword. It is also possible to build a new graph using the CONSTRUCT keyword. SPARQL's syntax is similar to that of the SQL language. The query below extracts all the pairs where *?subject* is a subclass of *?object*:

```
SELECT ?subject ?object
WHERE {
  ?subject rdfs:subClassOf ?object. }
```

However, as SPARQL makes the join operation implicit, it bears some resemblance with Prolog as in this query:

```
SELECT ?subject ?config
WHERE {
  ?subject rdfs:subClassOf <#FinishedProduct> .
  ?prop rdf:type owl:ObjectProperty .
  ?prop rdfs:range ?config . }
```

SPARQL is becoming a de facto standard for RDF. It is a stable language with quality implementations from various sources. Competitors include XQuery, a generic XML query language, which has not gained acceptance in the RDF community.

SWRL While SPARQL enables a programmer to extract information from an ontology, it is only designed for RDF. In addition, it cannot easily derive logical consequences from its results. To exploit fully the ontology knowledge, one needs a reasoning or inferencing mechanism. This is the purpose of a language like the semantic web rule language, SWRL [101]. SWRL rules have a Prolog-like structure. They consist of an antecedent corresponding to a conjunction of conditions (predicates) and a consequent. When the conditions are true, the consequent is also true and can be asserted. In addition to being an inference language, SWRL features an extension that lets it act like a query language, SQWRL.

SWRL is supported from the 3.4 version of Protégé in the form of a development environment with editing tools. This means that we can write, modify, and to a certain extent validate rules. However, version 3.4

is still in the beta stage at the time we are writing this paper. In addition, Protégé does not include a full-fledged inference engine. This means that it cannot by itself execute the rules. It just supplies a bridge that connects to an external module. So far, Protégé supports only one inference engine, Jess [102].

JastAdd JastAdd is not a query language in itself, but a general compiler construction tool with some very useful features; aspect oriented programming and attribute grammars. Using results from earlier work [103] we can automatically create a parser for an OWL ontology. Utilizing the aspect-oriented feature of JastAdd, we can then implement queries in the form of aspect modules that will be weaved in with the generated parser at compile time.

While it does not possess the expressive power of SWRL, it will enable the user to extract almost any information from the ontology with just a few lines of Java code.

Prolog Prolog – or Datalog – is a last example of reasoning tool that could be used to extract information from the ontology. Some Prolog implementations have a RDF interface like SWI Prolog that has been used with success in semantic web applications [104]. It is then possible to query an ontology from a logic program and to run inference rules on the result.

As Prolog predicates and rules have much in common with SWRL, logic programs written in Prolog and SWRL would be very similar and with equivalent performances. Difference would come from the location of the bridge between the ontology and the inference engine, at the RDF level for Prolog, at the OWL level for SWRL.

However, although Prolog is more expressive than other languages and has a proven record of industrial applications, Protégé does not support it. It is not standardized within the context of the semantic web either. This makes its choice, at the moment, riskier than the other options.

From an Ontology to the Appconf Sketch

The first step of the conversion pipeline generates the XML Appconf sketch from the ontology. As the SWRL formalism is more flexible and powerful, as well as adopted by the semantic web community, we are using it for this step in the demonstration prototype. As inference engine, we are using the built-in bridge that is for now only coupled to Jess.

However, SWRL is a new feature of Protégé 3.4 and although it already supports many *abox* and *tbox* built-in predicates, it is still under active development. Many of the predicates are not yet implemented. The beta

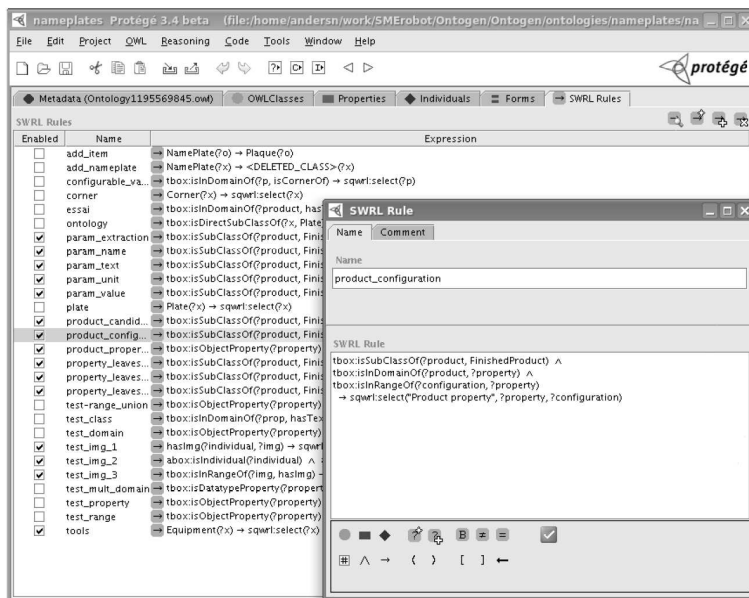


Figure 9.4 A screenshot of the SWRL interface in Protégé and an example of a rule.

version status of SWRL pose timetable problems and we are also using SPARQL to query the ontology and write the rules.

We wrote rules using both formalisms to extract information from the ontology. We show below an example of SWRL rule that finds all the properties of the subclasses of FinishedProduct and Figure 9.4 shows a screenshot of the editing window in Protégé. We also show its counterpart in SPARQL. We embedded the rules in the Java prototype using the Protégé API that resembles SQL drivers.

```

PREFIX list: <http://jena.hpl.hp.com/ARQ/list#>
SELECT ?product ?configuration
WHERE {
?product rdfs:subClassOf <#FinishedProduct> .
?property rdf:type owl:ObjectProperty .
{{?property rdfs:domain ?product} UNION
{?property rdfs:domain ?union .
?union owl:unionOf ?list .
?list list:member ?product}} .
?property rdfs:range ?configuration}

```

From the Appconf Sketch to Input Modalities

The second step of the conversion pipeline generates user input interfaces from the XML Appconf sketch. As final products frequently need to be customized according to the order, the manufacturing operator will be able to enter a part of the specifications at production time. In the demonstration prototype, we will investigate three configuration modalities that are core to the SMErobot project [93], namely E-forms, gestures, and spoken dialogue.

We are developing tools to generate automatically the work piece production forms, the gesture tracking and interpretation module, or the dialogue specifications from the XML Appconf. Before the piece is manufactured, the operator fills in the remaining data corresponding to the final piece using the modality of her/his choice. To ease the interaction, we are investigating a framework to combine simultaneously the different modalities so that the operator can use speech and gestures at the same time for instance.

Transformation Language As transformation language to produce the forms and the dialogue specifications, we are using XSLT. XSLT enables to apply transformations to Appconf nodes accessed via the XPath language. It can produce XML documents in formats like XHTML for the forms or VoiceXML for the dialogues. In addition, we are investigating the XSL-FO page-formatting standard where the description of a document content uses objects such as blocks, tables, footnotes, etc. It is richer than HTML-like descriptions and can be converted to PDF. The conversion of an XSL-FO document to a PDF uses a sequence of transformations that builds the XML tree, produces graphical objects, renders the objects as text areas with their pixel positioning, and finally generates PDF.

VoiceXML The demonstration prototype includes a voice modality that enables the operator to configure the product through a dialogue and hence have his hands free while he/she fills in the manufacturing options. The dialogue uses a system-initiative scheme, which means that the dialogue structure resemble a formfilling procedure where the user answers questions posed by the system.

We have chosen the Voice Extensible Markup Language, VoiceXML, to generate the dialogues. [105] is markup language that enables a programmer to build form-based, goal-oriented dialogues (system-initiative mostly). The user fills fields in forms using speech, where the field input can be constrained with a grammar. It is designed to be integrated in a speech server and supports IP telephony. As VoiceXML is a standard, the programs should be portable to any platform that supports this language.

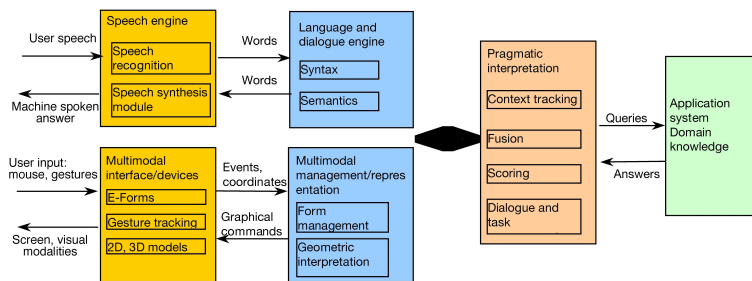


Figure 9.5 Multimodal dialogue architecture.

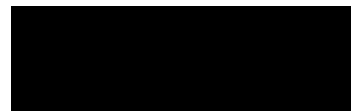
Perspectives: Multimodal Management

We will examine possible designs for the multimodal management architecture such as the one shown in Figure 9.5. It is mainly derived from a previous work of a member of the LTH team [106, 107] and a recent system developed by the Bell labs [108, 109]. One input channel corresponds to the speech recognition module, which transcribes the user's speech into a word stream. The language engine then processes the character flow dealing with syntax, which is constrained by the VoiceXML structure, and semantics. The semantic module converts words into a semantic representation that is common to speech and other types of interaction. The other channel corresponds to form filling, which are processed by the multimodal manager. The pragmatic module merges data from both modalities and keeps track of the context and the application goals. The resulting answers are either converted into speech to the user by a speech synthesizer or presented visually by a visualizing module.

The multimodal architecture will use a client-server architecture and instantiate some of the modules shown in Figure 9.5. We are currently defining them. In the future, it could evolve into an integration platform enabling partners to plug their applications. As a use case we consider interactions where the user fills in the data using one modality, the corresponding client sends the data to the server, and the server updates the context of all the modalities. There are then continuous visual or audio updates of the current context. For instance, an audio message is synthesized each time the user has selected an option with the form, to confirm or remind the next actions. Modality switching could be carried out manually by the user or automatically.

9.5 Conclusions

We have described the design and implementation of an assistive infrastructure based on the use of an ontological network to encapsulate knowledge on the product data and manufacturing processes. We have implemented a prototype ontology that serves as the main data source to automatically generate digital forms and voice dialogues to configure a wood nameplate manufacturing process. As a perspective, we intend to synchronize modalities for a more flexible, efficient user input. A first prototype will be available for demonstration in Spring 2008.



10

Service Oriented Architecture for Automatic Planning and Programming of Industrial Robots

Automatic engineering tool integration for task planning

10.1 Introduction

A service oriented architecture (SOA) has been developed based on a small domain specific ontology used for goal-driven automatic planning to produce a final robot program. The SOA services are orchestrated into work flow possibly containing partial operator input where needed, such as calibration routines. An overview of current research problems for incorporating web services in manufacturing is given in [110] and [111]. The W3C submission for semantic services description called OWL-S has been used in this work [112] and [113].

Constraint logic programming for automatic composition are seen in several works. In [114] finite domain constraints are used for encoding services for automatic discovery and composition. [115] utilizes constraints for automatic composition in the domain of security services (for example digital signing). In the work presented in this paper, the automatic composition and mediation are performed in a similar way by formulating OWL-S service models as finite domain constraint problems, but utilized within the manufacturing domain. The JaCoP system is used as a solver [116].

The idea of coordinating workflow in industrial settings have been

visited, although on a somewhat higher level than the robot cell [117], [118]. [119] discusses SOA architectures from a third-party perspective. Mediation and configuration of dataflows is visited in [120]. Ranking of services is touched in [121].

[122] discusses the “Device Profile for Web Services” proposal as a set of web standards suitable for incorporating factory elements in a SOA architecture, as does [123]. [124] looks at DPWS from a real-time perspective. [125] proposes semantic enhancements to existing automation standards (IEC TG65) as an attempt to standardize device descriptions. Though not touched further in this chapter the DPWS standard could prove to be the way of incorporating physical devices into workflows such as described here.

The method used in our work is based on a semantic service-oriented approach for automatic integration of process-oriented and vendor-specific task planning applications which in our work was selected to be robotic arc welding. Domain-specific reasoning simplifies creation, configuration and execution of task generation workflows involving complex modelling, simulation and planning software. From an industrial point of view we address the integration of important part in industrial automation such as automatic path planning and robot program generation, calibration routines and deployment issues to move programs from a simulation environment of a real physical environment.

The integration of these services creates a work flow of activities which increase the overall efficiency with respect to both productivity and quality. From a practical point of view, the programming time might be significantly lower compared with manual programming and even compared with robot simulation systems. Our approach will generate programs fully automatic which, if succeeded, reduce programming faults and generate a more consistent quality in the resulting program.

Challenges include the configuration and integration of the task planning software with other software needed to produce the resulting uploaded robot program in the robot controller. The integration should meet requirements related to efficient work flow and ease of operation by the user. However, the approach presented here provides some important benefits. An appropriate domain specific software for a given problem can be used and the goal-directed search for a valid task generation uses automatically building and negotiating semantically valid data flows, work flows and proper application configurations. Flexibility is achieved by allowing partial execution, a mix of off-line and on-line applications and tuning by the operator to override automatic methods where needed.

The ability to adapt to specific situations and calibrate a priori models to physical reality is crucial to minimize down-time during change-over between production batches or individual produced products. Fast, re-

liable and affordable calibration tools and methods has been developed and evaluated. The involvement of process models accessible during robot process specification and execution are important in order to change the focus from today's operator intensive robot-oriented programming phase to a task and process oriented programming which, if data are correct, is able to generate correct robot program from a model based specification. It is here important to be able to include typical process knowledge by the operator or producing company into a task strategy or rules for how the task should be generated to assure high quality without losing the possibility to trace and maintain process data.

The concept for automatic programming developed includes a deployment tool that support and help users to move the cell description and program from the off-line simulation to the robot controller and lead the user through the set-up of the robot cell based up on information gathered in the off-line work. The task generation is supported by a work flow within a Service Oriented Architecture (SOA) which supports automatic program generation but also allows the operator (user) to interact as needed within the programming process. The work presented here has been validated through simulation and verified by testing the actual generated robot programs. Tests verified the functionality and feasibility of the approach to generate robot programming automatically.

10.2 Experimental Platform

The application used in our development is robotic arc welding which represents an important process and application in industrial robotics. It also highlights issues related to process control and the importance of calibration of the weld gun and work object locations to achieve a defined quality of the weld.

A test bed has been used in the development and validation of the work of the automatic programming which was an important part in the experimental verification of the concept for integration of the different services as described in this paper. The aim was to show the feasibility of the described SOA methodology and its applicability, through the services provided and developed, to generate task programs for the robot in an automatic way.

The test bed consists of an ABB IRB140 robot, a work table with a flexible work piece set up with plates arranged in shapes typically found in arc welding of large constructions. The set up is shown in Figure 10.1, which also includes a welding gun (Binzel 22 degrees) and a laser displacement sensor for calibration purpose of the work objects.

The programming tools for the robot are Robot Studio from ABB Robotics

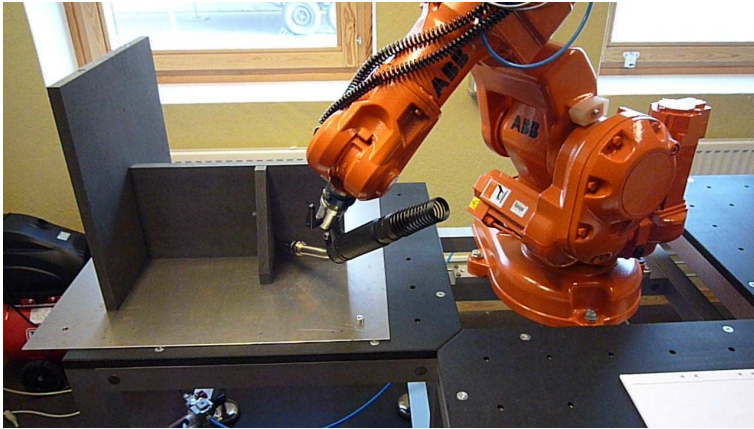


Figure 10.1 Test bed used in the development and verification of automatic programming.

which is the normal deployment tool for programs produced for ABB robots, RinasWeld from RINAS or 3DCreate from Visual Components or a combination of these for planning and generation of proper weld operations. In the development of automatic programming, services are set up and defined for the software tools which are integrated in a work flow to produce different parts of the program to be executed on the robot. For the test bed, the programming methodology is validated using the normal operator tools for the robot such as the direct on-line programming using the teach pendant and the off line and simulation environment using Robot Studio.

RinasWeld is proprietary software that specifically in the domain of arc welding is able to plan and produce robot programs for welding operations. These are generated with respect to collision avoidance, singular areas, joint limits and specific process related parameters set up for the specific welding operations. Fine calibration is included in the generated robot program that typically is a series of search movements to detect the plates by touching them with the gas nozzle or the tip of the weld wire. This routine was redefined in this work to allow for the laser displacement sensor described below. Input to the system are models of robot, work set-up and work pieces to be welded included weld joint parameters for the welding process. A screen shot from the software can be seen in Figure 10.2 (left).

In our work, it was assumed that 3D models of the work piece was known before hand. However, in most cases, a calibration is needed to compensate for displacements of the work piece with respect to the robot.

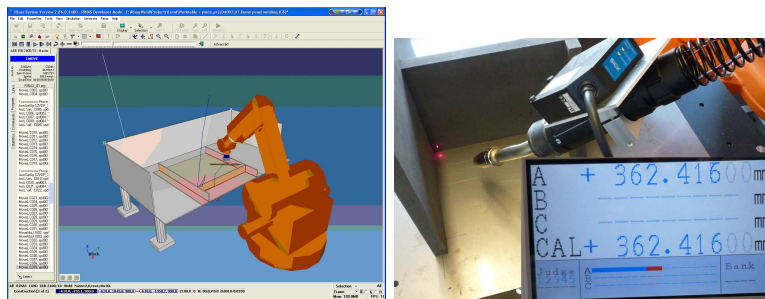


Figure 10.2 RinasWeld screen shot during simulation and path planning (left), Welding gun with displacement sensor attached (right).

This is due to tolerances associated with large constructions or products in low volume, but also geometrical displacements occurring during welding. Thus, sensing capability is needed both before and during task execution for fine calibration of the workpieces. Work related to global calibration has been developed and verified earlier using different tools such as vision and calibration using the robot itself producing a rough estimate of work objects within 10 mm of the nominal location. Fine calibration is developed as an integrated part within RinasWeld.

Standard test workpieces have been developed for the test bed and are used in the development and validation of the automatic programming methodology. The welding gun is used to simulate welding and include as sensing capability a laser displacement sensor. The laser displacement sensor is mounted on the welding gun to measure the distance to the weld plates and can be used to measure the same feature of a plate as a probing method used on most arc welding applications. With the displacement sensor, the measuring distance is typically 350 mm and the measurement can be done much faster utilizing smaller motions of the robot. By introducing new search methods in the software which produce the robot programs, the programs are not only produced fast, they will execute faster as well. This is an example where the automatic programming methodology will not only produce programs faster, but also with new features which speed up the execution in certain situations, see Figure 10.2 (right) which shows the welding gun with laser displacement sensor.

The methodology to plan and generate programs automatically has been validated and proved successful where programs were generated as RAPID program code which is the native language for the ABB robot controller, and then manually deployed to the robot using Robot Studio. For the continued work, the SOA architecture will support a workflow

which automatically generate and search for services to move a generated program to an executable robot code as described here.

The most commonly used practice to localize the weld joint is to apply a set of search movements of the welding gun where the tip of the weld electrode is forced to touch the plates. By using this technique a weld joint can be localized in a way which is sufficient for the welding process. However, if used extensively, the time for the procedure may be in the order of 20-30 percent of the available productive time. To speed up this process, several methods can be applied which make use of general vision based sensors. However, such methods are in many cases unsuitable due to the process environment or the workpiece set-up, or changes may occur during the processing of the workpiece which is difficult to measure by the sensors. Alternative solutions are in such cases to use a sensor mounted on the robot which moves with the welding gun. In the general case a seam tracker can be used for this purpose which uses a laser source and triangulation of a scanning laser stripe transverse to the weld joint (or plates, edge, etc) to measure the distance profile of the joint or plate measured.

These sensors are standard industrial devices but show a few drawbacks: they are rather expensive to use, add complexity to the system and occupy a geometrical space at a certain place in front of the welding gun. For this reason they cannot be said to be a general solution to the problem presented. Another solution is to apply a specific dedicated sensor to measure the properties of the plate or joint. One such sensor which can be used is the CSS-WeldSensor from Oxford Sensor Technology which applies a precise scanning of a surface area by rotating the laser head and applying triangulation to measure the distance in the same way as a seam tracker, but without the scanning techniques [126].

The method is efficient and our approach can be said to be a variant of this but in our case we are using a simple laser displacement sensor and the movements are performed by the robot producing a more versatile and low cost system. As with the CSS-WeldSensor, the laser displacement sensor can be mounted off-set to the weld gun at any suitable place and provide a rugged solution to increase efficiency and quality of the work process of the robot.

10.3 Deployment of Programs

To generate a program automatically is not enough for being able to actually run the application in the robot system. For this purpose, tools that support in the deployment process of the automatically generated program to the physical system has been developed and integrated in the

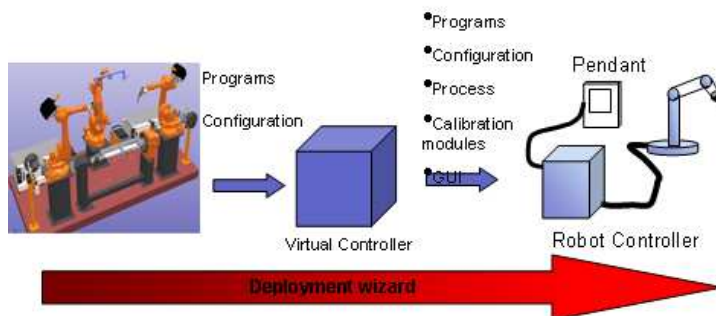


Figure 10.3 The developed Deployment Wizard is a tool to guarantee a complete and correct set up of a robot system from a simulation environment.

work flow, see Fig. 10.3.

Deployment of programs is often a neglected problem and from a practical point a very difficult problem which has been a hindrance to the adoption of offline programming in industry, especially in improving productivity for small and medium enterprises which rely on small batches and new products. Thus there is a need for tools and methods to help users of off-line programming systems to deploy off-line solutions to the on-line robot system.

The method and a tool described here address the off-line to on-line programming calibration issues. That is when a user has programmed or adjusted a program in a 3-D off-line environment and then wishes to take that program to the factory floor.

The problems have been many, among which is that after building the program using CAD models or work station models; these stations must then be calibrated to match the physical robot station. This typically is done by using the robot as the measuring device; the user jogs the robot to at least 3 points to calibrate a work object, or reference frame.

When there are many stations or many objects, the user must go to all of the objects and jog to many points. Typically the user must write down on a piece of paper all of the objects to be calibrated or somehow manually mark in the downloaded robot program the frames, objects, and tools that need to be calibrated on the floor using the robot. This takes time and is prone to error. There is in general no help for the user to remember the points, the objects calibrated, nor help in visualizing the steps necessary. The user has no visual clues when working on the Teach Pendant and has to remember in his/her head what the objects in the 3-D simulation

were. While the standard calibration tools present in the robot controller and teach pendant work, there is no customized calibration that would help the user directly with the task.

However, all information needed in this process to help the user is already available in the off-line 3-D system, but not available on the controller and teach pendant when working on the factory floor in the robot cell. In this work, a configuration tool has been developed to support the operator in this process. This tool uses the graphical layout of the robot cell in the simulation program to generate a system for the Virtual Controller with the correct configurations and options. This means that the user, by one command, can generate a completely new robot system in just a few seconds and that this system contains correct configuration and placement for robots, tracks, positioners, etc.

This tool will also prepare the robot system to be easily configured when being deployed to the real online robot cell by adding calibration support. The tool is designed to react on reconfiguration suggestions from the user or from an external decision tool to an existing robot cell and will provide setup support for process and equipment work cell changes. The Deployment Wizard will then identify these changes, generate the needed cell setup and help the user through all the steps necessary to calibrate the work object, frames, and tools used in the cell.

This is accomplished by taking 3-D snapshots from the offline simulation tool, and combine these with a Teach Pendant Application Program which presents the images on the teach pendant and guides the user step-by-step through the calibration process. In addition, this is combined with a robot program which can be used to quickly move the robot to all of the entered points (for calibration checks) and to save the points for future verification. This function in RobotStudio is called “Create System from Layout”. The “Create System from Layout” function allows the user to easily test different station configurations and to simply reconfigure the actual cell layout.

For a user with small batches that need that the production cell to be reconfigured frequently this tool will provide a useful support and the function analyses the cell and the robot program and generates the following output:

1. The robot program(s) for calibration contain the robot motion instructions and robot target storage to move the robot to the necessary calibration positions that are needed for calibrating the different frames and work objects in the cell.
2. The 3-D snapshots are pictures, taken from the 3-D simulation, which are automatically generated and scaled to fit on the Teach Pendant. These pictures show the objects and frames to be cali-

brated and what points are to be used when jogging the robot and calibrating.

3. The Teach Pendant Application program is a .NET assembly that is generated based upon the data in the 3-D simulation and which contains the step-by-step user interface that shows the 3-D snapshots and guides the user through the calibration process. The application is essentially a “wizard” in that it makes the calibration process easy and straightforward. See Figure 10.4 for an example of the application running on the Teach Pendant.

Here the current calibration status is shown, if the frame is not calibrated the user can run the calibration sequence (or recalibrate, if needed). Depending on the method selected during the setup-phase a robot program starts, instructing the user to perform the calibration.

10.4 System Integration and Workflow Support

This section presents the SOA as an integration concept to increase the level of automation in non-trivial task generation processes that may incorporate several CAD/CAM/CAPP applications as well as different calibration equipment and product/process knowledge. The challenge is to increase system efficiency without adding complexity to the operator of the system with access to complex planning functionality without requiring expert application and/or integration knowledge. The proposed integration concept aims at achieving this by increasing the level of automation during creation and execution of task generation processes:

- Automate application interaction thus removing the need for detailed application operational knowledge.
- Automate information flow and message contents between applications removing the need for detailed application integration knowledge.
- Automate configuration of applications, for instance by assisting selection of process parameters from product knowledge.

The operator is presented with a simple user interface (at present a web search interface) that allows the operator to formulate the intent of the task generation process as a search query. A search is performed to find valid task generation processes among available applications, equipment, and processes. The search result is presented to the operator and the process can be directly executed by the press of a button.

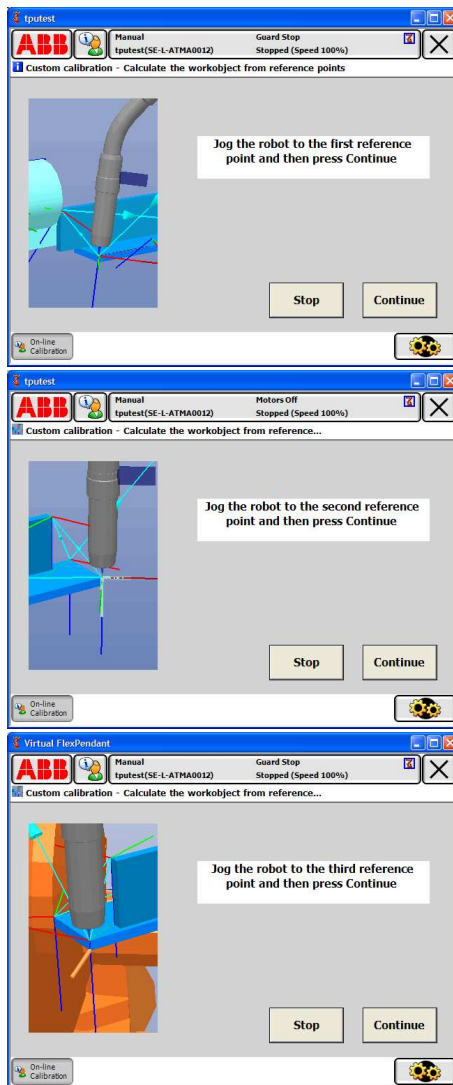


Figure 10.4 Calibration program with step-by-step guidance.

The integration concept uses results and techniques from service-oriented architecture, web services and semantic web:

- Application/equipment functionality is exposed as web services to enable automatic invocation of functionality.

- Semantic descriptions provide machine-readable explanations of web service functionality and invocation to enable automatic search for web service orchestrations (task generation processes).
- Ontology-stored concepts provide a machine-readable common understanding of web service functionality and message exchange needed to integrate several web services into a task generation process.
- Ontology concepts also provide an understanding of product, process, and application knowledge to allow inference of some web service message content (interpreted as application configuration) during web service orchestration.

Integration is performed in two stages. During the first stage (semantic stage) a constraint-logic reasoner performs web service matchmaking, web service orchestration, and mediation of web service message content based on semantic web service descriptions and a search query stated by the operator. A logic reasoner infers message content based on ontology-stored knowledge. The outcome of this stage is one or several process execution scripts representing valid task generation processes. The script describes a task generation process by means of containing references to all participating services, containing a complete data flow graph between services, containing a complete work flow graph between services, containing configurations of data flows between services and containing configurations of services themselves. Execution state is part of the script allowing the operator to save the current state and resume later, and also re-planning part of the script. During the second stage (execution stage) the script is executed. Execution state is persisted allowing the SME operator to retain a log of the task generation process.

The prototype built for validation purpose illustrates the concept. The operator generates a weld task for the workpiece and cell equipment described in the test bed by performing a search in a web interface. The search result orchestrates a planning tool (RinasWeld by KPS Rinas) with a simulation tool (RobotStudio with Deployment Wizard by ABB) and product CAD into a task generation script that ends with task deployment to the test bed robot controller (uploaded RAPID executable robot code).

Revisiting the task generation process of the test bed scenario in more detail the central tool used is the RinasWeld planner which generates weld programs. It needs to be fed with workpiece geometry, enough information about the target cell to be able to create the cell in the planner (equipment), and calibration data for cell and workpiece. The planner also needs to be configured by selecting proper welding process parameters (such as leg length), and selecting welding strategy options depending on, among

others, workpiece geometry, workpiece material properties, and type of weld gun. The other tool in the scenario is the ABB RobotStudio simulation tool. The assumption is that a model of the cell is maintained in the tool. Information about equipment and calibration data (equipment positioning, work frames, and tool frames) can then be extracted and used by the planning tool. The deployment wizard functionality ensures that calibration data is up-to-date with the physical cell. The third tool used is a generic CAD software mock-up capable of generating workpiece geometry compatible with the planner tool. Meta-information about the workpiece can be used to infer process parameters for the planner tool.

Taking a bottom-up approach the implementation of the prototype consists of the following parts:

- Expose application interfaces as web services.
- Expose task generation processes as process execution scripts containing data, work flow steps and execution state.
- Implement/find a process execution engine to execute the script.
- Create a common body of semantic knowledge shared between applications (keeping it small and focused on task generation)
 - Expose descriptions of application interfaces as ontology-based process descriptions.
 - Expose web service messages and message content data formats as ontology-based descriptions.
 - Expose product, process and application knowledge as ontology-based descriptions.
- Use a logic reasoner to derive process-specific parameters (static message content) from the body of product/process/application knowledge.
- Use a finite-domain constraint-logic reasoner to synthesize valid process execution scripts by composing web services and mediating message content from the ontology descriptions.

The integration concept was bootstrapped and evaluated using available service-oriented technology where possible, such as de-centralized software services (DSS) from Microsoft Robotics Studio, OWL ontology language with the OWL-S upper ontology was chosen as basis for semantic service descriptions and storage of product/process/application knowledge.

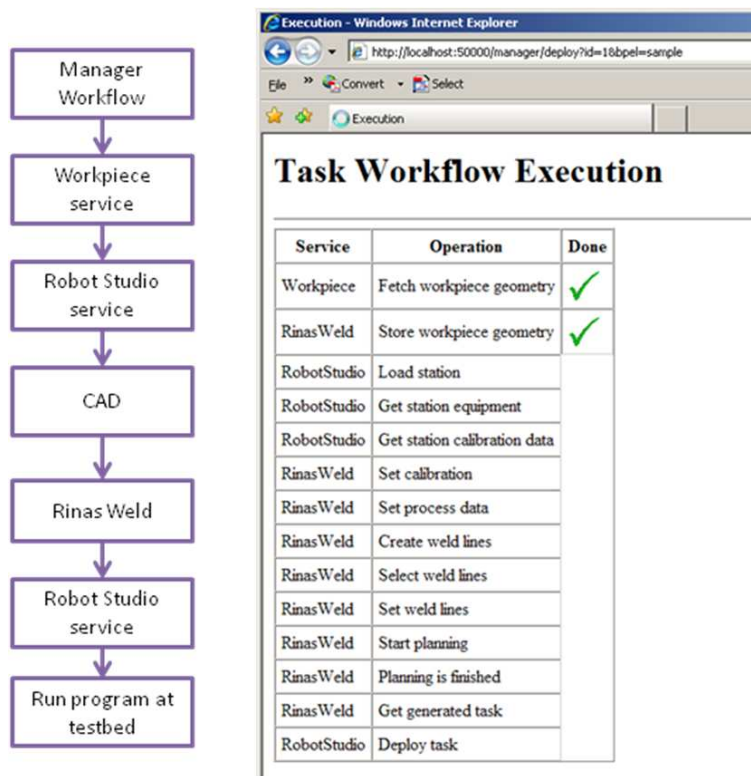


Figure 10.5 Task workflow during execution and generation of the robot program.

In conclusion, the prototype generates a work flow involving RobotStudio for calibration information, RinasWeld for planning of tasks, and a sample CAD provider, see Figure 10.5. The operator specifies intent in a web based search query interface. The query is transformed into a process script (workflow, dataflow) through the Protégé and JaCoP tools performing reasoning and constraint-based search on semantic descriptions of the available services. A taskgeneration-specific ontology contains common concepts used in the service descriptions necessary for transforming a query into a process script. The script is interpreted and executed by a manager service handling the message flow between services (RobotStudio Deployment Wizard, RinasWeld, sample CAD provider).

A problem was handling of operator interaction within the automatic script generation and execution. Both participating applications needed dialogues to be part of the script work flow (RinasWeld for selection of weld lines, Deployment Wizard for confirmation of calibration status).

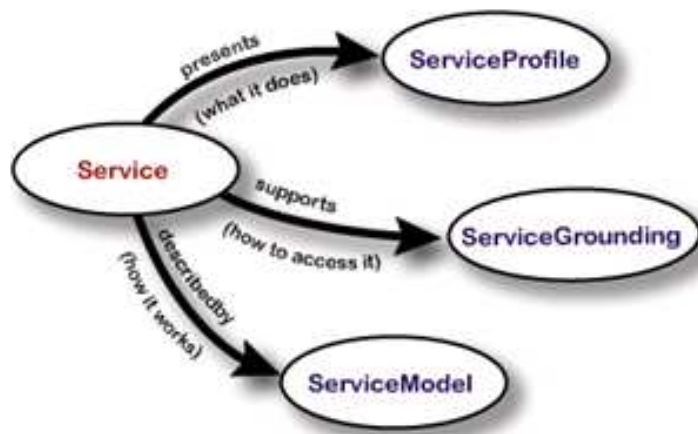


Figure 10.6 Excerpt from the OWL-S standards proposal. Overview of the OWL-S service description.

The solution for the prototype was to include dialogue messages as part of the application web service, displaying a dialogue with message-specific content when executed.

10.5 OWL-S Extension for Constraint-Based Search

OWL-S considers a service description to consist of three parts: The service profile provides a classification of the service, easing the job for discovery algorithms. The service model contains a process specification of service, providing a local data and work flow model. The service grounding provides a definition of concrete message formats for communicating with the service, currently the web service description language (WSDL) is supported, see Figure 10.6.

Figure 10.7 shows the top-level concepts of OWL-S as seen in the Protégé tool. The concepts are separated by namespaces (shown as prefixes followed by colons). The major OWL-S concepts are presented in the service namespace. The other namespaces present concepts used for further describing the major concepts, in particular the process namespace describes control flow and binding constructs used for constructing work and data flow graphs.

Two extensions to the OWL-S ontology have been found necessary. Applying automatic composition and mediation techniques for configuration

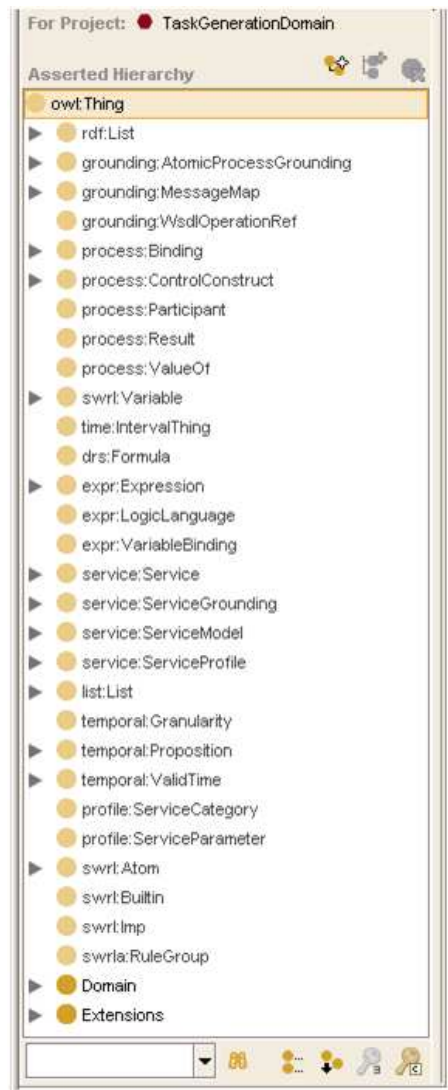


Figure 10.7 The OWL-S top level concepts as viewed from the Protégé tool.

are most likely to produce no solution or many solutions. Therefore there is a need to rank solutions for the system to suggest a preferred solution and maybe sort the remaining solutions to be able to provide the operator with alternative solutions. The second extension regard configuration of services and data flow. OWL-S does not offer means to express several

alternative complex data transfers, such as a service having the ability to transfer geometry in a variety of file formats. The same can be said for complex configurations of services, such as configuring strategies for task generation. The extension adds a configuration attribute to parameters and services. This attribute may contain a description of capabilities and requirements. The OWL-S process modeling language is re-used as a configuration description language.

The OWL-S ontology needs a companion ontology with domain-specific concepts to describe services. Figure 10.9 shows a snapshot of the task generation ontology being developed for the prototype test bed. The ontology identifies different roles in the task generation process, covered below the service profile concept. The ontology also defines ranking values, declared below the ranking value concept. The high-level data transfers that may occur in a task generation are identified below the connection profile concept. Configuration options are identified below the configuration model concept, currently showing only example options for geometry exchange. The data value concept is a container for data generated during execution.

Basic enhanced service description Service descriptions are created by instantiating concepts from the OWL-S and domain ontologies. Figure 10.10 shows the description of the 3DCreate tool from Visual Components, acting as a CAD provider. The viewing tool is OWL-S editor, a plug-in for Protégé. This editor consists of four windows to the left, showing (top to bottom) services, service profiles, service models, and service groundings. The service being described is called `RankedService_3DCreate_CADProvider`. It has the `CADProfile` profile, is described by the `CompositeProcess_3DCreate_CADProvider` service model, and sends one message (called CAD) that is grounded to a WSDL description, `Wsd1Grounding_CAD`. To the right is shown the properties of the service concept, which has attributes referring to the profile, model, and grounding of the service. It also has the extension attribute `hasRanking` ranking the service, here for the level of manual work needed to execute the method/procedure the service represents.

Figure 10.11 shows the service model for the 3DCreate CAD provider service. The service model is a composite consisting of a sequence featuring the sending of a CAD message. The `AtomicProcess_CAD` is modeled as an atomic process with one output parameter, `ConfigurableOutput_CAD`. This message is sent routed internally to a corresponding composite message through the `Produce` construct. The graph to the left in the figure shows both the work flow and the data flow, the work flow using beige color and the data flow blue color.

The CAD message is configurable (i.e. not semantically grounded).

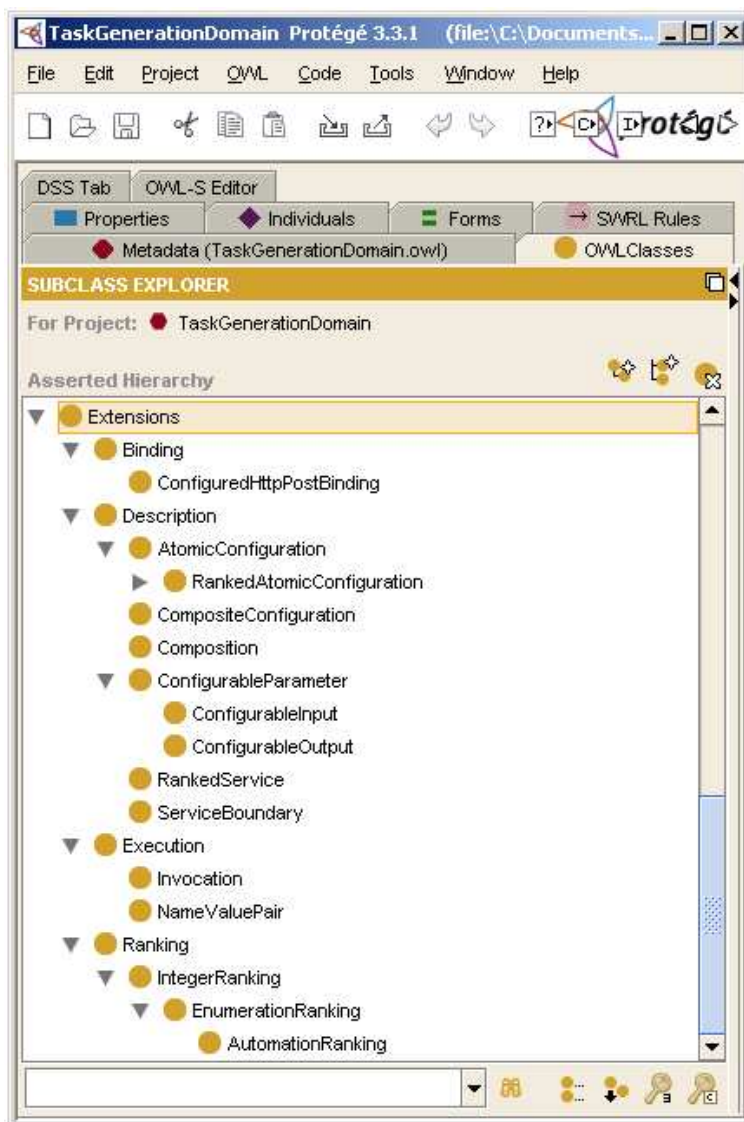


Figure 10.8 Extensions to OWL-S for declaring service and parameter configurations and ranking services and configurations. The composition, service boundary and name value pair are deprecated concepts.

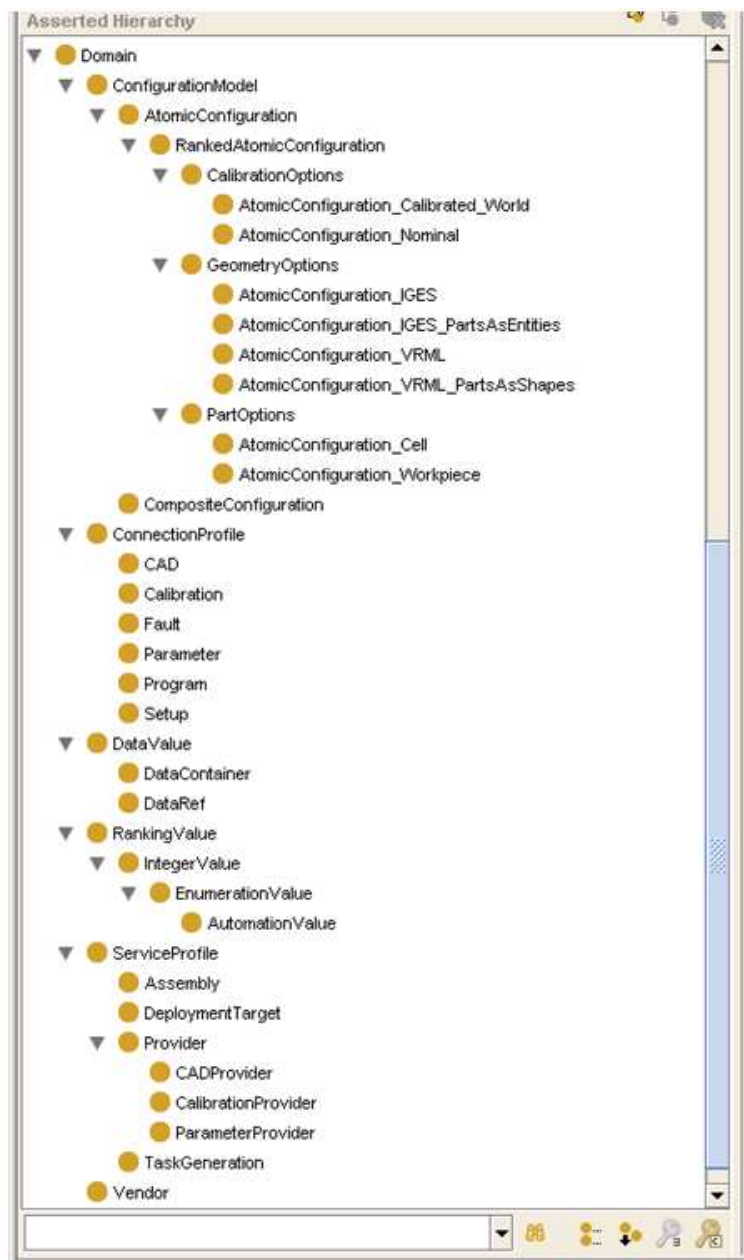


Figure 10.9 Domain-specific ontology.

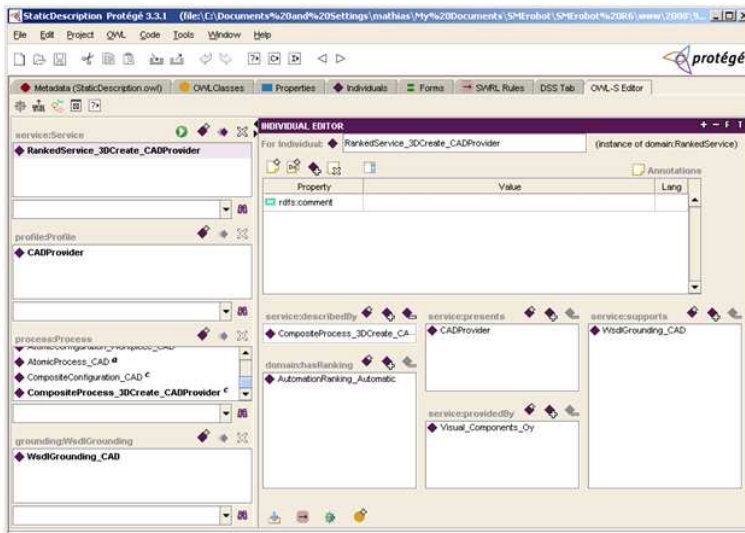


Figure 10.10 The service description for Visual Components 3DCreate application in the role of CAD provider.

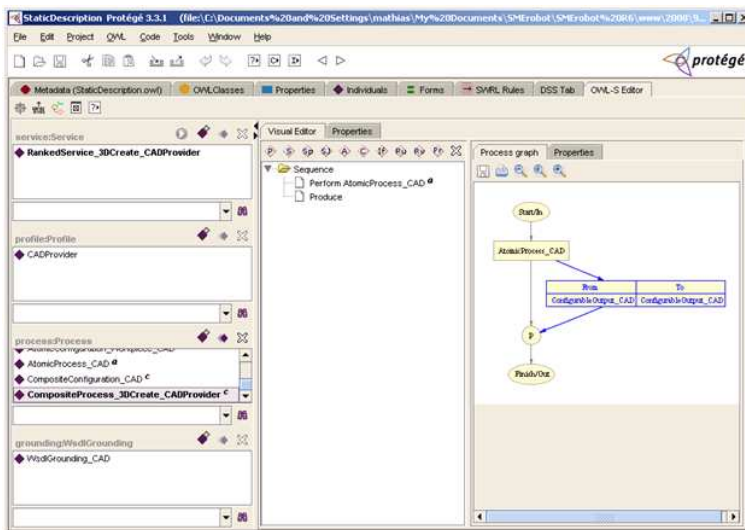


Figure 10.11 Service model.

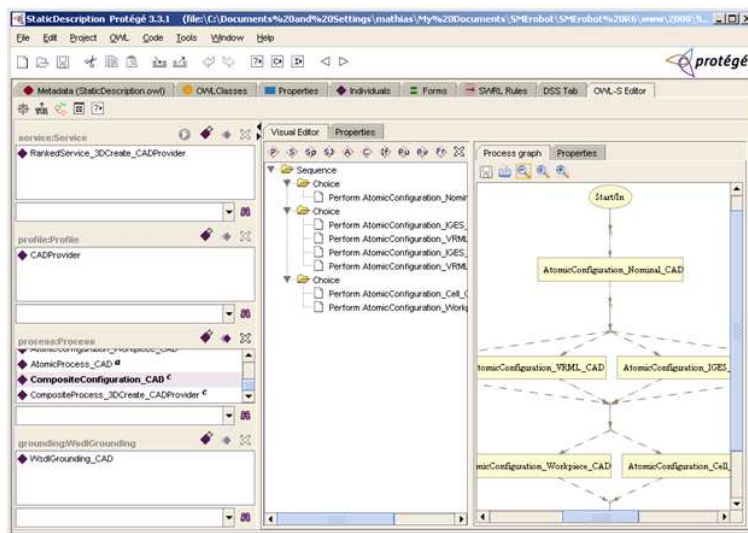


Figure 10.12 The configuration graph of the CAD message.

Figure 10.12 shows the configuration graph for this message. Right now it consists of a sequence populated with possible choices. More complex configuration scenarios are possible to model since the configuration language is using process control constructs. But this is not supported in the reasoning mechanism.

Constraint-Based Search using JaCoP

Automatic composition and mediation of are performed by formulating OWL-S service models as finite domain constraint problems. The JaCoP system is used as solver [116].

Composition For composition, the goal is to arrive at fully connected work and data flow graphs. This is achieved by searching for message exchange sequences requiring fully connected work and data flow graphs, possibly ranking solutions.

An example illustrates the approach. The example searches for all valid compositions starting from the intent, given as starting constraints. Solutions are not ranked for brevity. Available services and connections are encoded as finite domain values:

```
// Services
private static final int TEST_NOSERVICE = 1;
private static final int TEST_SERVICE1 = 2;
```

```
private static final int TEST_SERVICE2 = 3;
private static final int TEST_ALLSERVICES = 3;

// Connection types
private static final int TEST_NOTYPE = 1;
private static final int TEST_TYPE1 = 2;
private static final int TEST_ALLTYPES = 3;

// Service outgoing connections
private static final int TEST_NOOUTGOING = 1;
private static final int TEST_SERVICE1_TYPE1_OUT = 2;
private static final int TEST_ALLOUTGOING = 2;

// Service incoming connections
private static final int TEST_NOINCOMING = 1;
private static final int TEST_SERVICE2_TYPE1_IN = 2;
private static final int TEST_ALLINCOMING = 2;
```

The search is declared as a finite domain problem in the JaCoP syntax. A message exchange is encoded using five finite domain variables. Two for describing the origin of the message (service and connection), two for describing the target of the message (service and connection), and one for describing the connection type. A workflow consists of an array of these five variables, the example allows a maximum size of three sequence steps.

```
public void test_allValidCompositions() {
    int size = 3;

    FDstore store = new FDstore();

    FDV[] fromService = new FDV[size];
    FDV[] fromConnection = new FDV[size];
    FDV[] toService = new FDV[size];
    FDV[] toConnection = new FDV[size];
    FDV[] type = new FDV[size];
    FDV[] used = new FDV[size];

    // Initialize FDV domains
    for (int i=0; i<size; i++) {
        fromService[i] = new FDV(store, TEST_NOSERVICE, TEST_ALLSERVICES);
```

```
    toService[i] = new FDV(store, TEST_NOSERVICE, TEST_ALLSERVICES);

    type[i] = new FDV(store, TEST_NOTYPE, TEST_ALLTYPES);

    fromConnection[i] = new FDV(store, TEST_NOOUTGOING,
        TEST_ALLOUTGOING);
    toConnection[i] = new FDV(store, TEST_NOINCOMING,
        TEST_ALLINCOMING);

    used[i] = new FDV(store, 0, 1);
}
}
```

Basic constraints ensure that a sequence is as short as possible, that a message is not sent from a service to itself, that a message is sent between two services, and that the sender and receiver connections are of the same type. Here can also be constraint for shortest solution (least number of steps), ranking, etc.

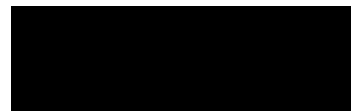
Mediation For mediation, the goal is produce a set of choices that is valid for services exchanging a message. Different choices can be ranked according to service preferences. The configuration options consist of set of choices, each consisting of a set of options. For each service connection the choices are expressed as a list of finite domain variables with the domain being the set of options.

10.6 Results

Practical tests during a verification phase have shown a consistent and reliable task programming which produced an uploaded program in the robot controller in less than one minute. The program was generated to perform arc welding operation based on a CAD model where up to five weld joints were selected followed by the automatic generation of work document and subsequent planning and robot program generation as discussed above. The weld paths was not considered as “easy” to program as they needed careful planning by the task planner to consider close to joint limits, configuration changes and singular areas, which was detected during manual programming and off-line programming using RobotStudio in the traditional way. Furthermore, the deployment issue related to calibration of work object and tool data as discussed above were also included in the final workflow operation.

10.7 Conclusions

SOA approach for automatic application integration with the goal of automatic task generation from CAD models has been developed. The design and operation of the automatic task generation was validated to successfully integrate the different services needed and generate a task program for upload in the robot controller. The result was demonstrated as part of the final demos in the SMErobot project.



11

Bibliography

- [1] P. I. Corke and S. A. Hutchinson, "Real-time vision, tracking and control," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, (San Francisco, CA), pp. 622–629, 2000.
- [2] K. Nilsson, *Industrial Robot Programming*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, 1996.
- [3] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. NJ: Prentice Hall, 1998.
- [4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley, 1993.
- [5] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the 7th IEEE International Conference on Computer Vision*, vol. 1, pp. 666–673.
- [6] J. Nilsson, *Real-Time Control Systems with Delays*. PhD thesis, Lund Institute of Technology, 1998. ISRN LUTFD2/TFRT-1049-SE.
- [7] B. Siciliano and L. Villani, *Robot Force Control*. Dordrecht, NL: Kluwer Academic, 1999.
- [8] D. Gorinevsky, A. Formalsky, and A. Schneider, *Force Control of Robotic Systems*. Boca Raton, FL: CRC Press, 1997.
- [9] T. Yoshikawa, "Force control of robotic manipulators," in *Proceedings of IEEE International Conference on Robotics and Automation*, (San Francisco, USA), pp. 220–226, 2000.
- [10] H. Born and J. Bunsendal, "Programmable multi sensor interface for industrial applications," in *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems*, (Baden-Baden, Germany), pp. 189–194, 2001.

- [11] A. Martinsson, "Scheduling of real-time traffic in a switched ethernet network," Master's thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, March 2002.
- [12] F. Calugi, "Observer-based adaptive control," Master's thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, April 2002.
- [13] <http://www.modelica.org>. Modelica.
- [14] S. E. Mattsson and H. Elmqvist, "An overview of the modeling language modelica," in *Proceedings of the Eurosim98 Simulation Congress*, (Helsinki, Finland), pp. 182–186, April 1998.
- [15] <http://www.dynasim.se>. Dynasim Inc.
- [16] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 103, no. 2, pp. 126–133, 1981.
- [17] J. Kruth, B. Lauwers, P. Klewais, and P. Dejonghe, "Nc-postprocessing and nc-simulation for five-axis milling operations with automatic collision avoidance," *International Journal for Manufacturing Science and Technology*, vol. 1, no. 1, pp. 12–18, 1999.
- [18] K. Nilsson and R. Johansson, "Integrated architecture for industrial robot programming and control," *Journal on Robotics and Autonomous Systems*, vol. 29, pp. 205–226, 1999.
- [19] H. Bruyninckx, "Open robot control software: The orocos project," in *Proceedings of the International Conference on Robotics and Automation*, vol. 3, (Seoul, Korea), pp. 2523–2528, 2001.
- [20] M. Summers, "Robot capability test and development of industrial robot positioning system for the aerospace industry," in *SAE 2005 AeroTech Congress & Exhibition*, (Grapevine, TX), 2005.
- [21] E. Dégoulange, P. Dauchez, F. Pierrot, and P. Prat, "Determination of a reference model for controlling the deformation of an industrial robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 343–351, 1994.
- [22] H. Kihlman, *Affordable automation for airframe assembly – Development of key enabling technologies*. PhD thesis, Linköping University, Linköping, Sweden, 2005. Thesis 953.
- [23] S. Van Duin, "A comparison of indoor gps versus laser tracking metrology for robotic drilling," in *Aerospace Manufacturing and Automated Fastening Conference and Exhibition*, (Toulouse, France), 2006.

- [24] S. Van Duin and H. Kihlman, "Robotic normalizing force feedback," in *SAE 2005 AeroTech Congress & Exhibition*, (Grapevine, TX), 2005.
- [25] S. Kawaji, M. Arao, and Y. Chen, "Thrust force control of drilling system using neural network," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 1, (Como, Italy), pp. 476–481, 2001.
- [26] G. Alici, "A systematic approach to develop a force control system for robotic drilling," *Industrial Robot: An International Journal*, vol. 26, no. 5, pp. 389–397, 1999.
- [27] W. Lee and C. Shih, "Control and breakthrough detection of a three-axis robotic bone drilling system," *Mechatronics*, vol. 16, no. 2, pp. 73–84, 2006.
- [28] ABB Robotics, *Application Manual – Force Control for Assembly*, 2005. Ref. 3HAC025057-001.
- [29] R. Johansson, A. Robertsson, K. Nilsson, T. Brogårdh, P. Cederberg, M. Olsson, T. Olsson, and G. Bolmsjö, "Sensor integration in task-level programming and industrial robotic task execution control," *Industrial Robot: An International Journal*, vol. 31, no. 3, pp. 284–296, 2004.
- [30] A. Blomdell, G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, A. Robertsson, and J. J. Wang, "Extending an industrial robot controller – implementation and applications of a fast open sensor interface," *IEEE Robotics & Automation Magazine*, pp. 85–94, September 2005.
- [31] T. Olsson, A. Robertsson, and R. Johansson, "Flexible force control for accurate low-cost robot drilling," in *IEEE International Conference on Robotics and Automation (ICRA07)*, (Rome, Italy), pp. 4770–4775, 2007.
- [32] A. Shiriaev, A. Robertsson, and R. Johansson, "Friction compensation for passive systems based on the lugre model," in *Proceedings of the 2nd IFAC Workshop on Lagrangian and Hamiltonian Methods for Nonlinear Control*, (Seville, Spain), pp. 183–188, 2003.
- [33] T. Olsson, *High-speed vision and force feedback for motion-controlled industrial manipulators*. PhD thesis, Department of Automatic Control, Lund University, May 2007. TFRT-1078.
- [34] T. Olsson, R. Johansson, and A. Robertsson, *Dynamical Vision*, vol. LNCS 4358/2007 of Lecture Notes in Computer Science,

ch. Force/vision based active damping control of contact transition in dynamic environments, pp. 299–313. Berlin-Heidelberg-New York: Springer-Verlag, 2007.

- [35] H. Kihlman, T. Brogårdh, M. Haage, K. Nilsson, and T. Olsson, “On the use of force feedback for cost efficient robotic drilling,” in *SAE 2007 AeroTech Congress & Exhibition*, (Los Angeles, CA), 2007. SAE Paper 2007-01-3909.
- [36] C. Natale, R. Koeppel, and G. Hirzinger, “A systematic design procedure of force controllers for industrial robots,” *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 2, pp. 122–131, 2000.
- [37] V. Lippiello, B. Siciliano, and L. Villani, “An open architecture for sensory feedback control of a dual-arm industrial robotic cell,” *Industrial Robot*, vol. 34, 2007.
- [38] H. Zhang, J. Gan, J. Wang, and G. Zhang, “Machining with flexible manipulator: Towards improving robotic machining performance,” in *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1127–1132, July 2005.
- [39] T. Brogårdh, “A device for relative movement of two elements,” 1996. Patent WO 97/33726.
- [40] <http://www.smerobot.org>, 2005. SMErobot.
- [41] H. Kihlman, G. Ossbahr, M. Engström, and J. Anderson, “Low-cost automation for aircraft assembly,” in *SAE 2004 Aerospace Manufacturing Automated Fastening Conference Exhibition*, (St Louis, MO, USA), September 2004. SAE Paper 2004-01-2830.
- [42] <http://www.gudel.com>. Güdel AG.
- [43] A. Robertsson, T. Olsson, R. Johansson, A. Blomdell, K. Nilsson, M. Haage, B. Lauwers, and H. de Baerdemaeker, “Implementation of industrial robot force control – case study: High power stub grinding and deburring,” in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2006)*, (Beijing, China), pp. 2743–2748, 2006.
- [44] <http://www.nortoncast.co.uk>. Norton Cast Products Ltd.
- [45] <http://www.beckhoff.com>. Beckhoff Automation GmbH.
- [46] L. Johannesson, V. Berbyuk, and T. Brogårdh, “Gantry-tau – a new three degrees of freedom parallel kinematic robot,” in *Parallel Kinematic Machines in Research and Practice; The 4th Chemnitz Parallel Kinematics Seminar*, pp. 731–734, 2004.

- [47] <http://www.python.org>. Python.
- [48] <http://www.visualcomponents.com>. Visual Components.
- [49] <http://www.rinas.dk>. KPS Rinas.
- [50] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.
- [51] M. Murray, G. Hovland, and T. Brogårdh, "Collision-free workspace design of the 5-axis gantry-tau parallel kinematic machine," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2006)*, (Beijing), 2006.
- [52] I. Dressler, A. Robertsson, and R. Johansson, "Automatic kinematic calibration of a modular gantry-tau parallel robot from a kinematics point of view," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA08)*, (Pasadena, CA, USA), pp. 1282–1287, 2008.
- [53] <http://www.castingstechnology.com>. CTI.
- [54] R. W. Atherton, "Moving java to the factory," *IEEE Spectrum*, December 1998.
- [55] M. Jern, "3d data visualization on the web," in *Proceedings of the 1998 MultiMedia Modeling (MMM98)*, (Los Alamitos, CA, USA), pp. 90–99, IEEE Computer Society, 1998.
- [56] G. Hirzinger, B. Brunner, R. Koeppel, K. Landzettel, and J. Vogel, "Teleoperating space robots – impact for the design of industrial robots," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE97)*, (New York, NY, USA), IEEE, 1997.
- [57] P. Krusell, "Den digital drömfabriken (in swedish; eng: The ideal digital factory)," *PLASTForum*, vol. 3, 1998. Sweden.
- [58] Deneb Robotics, Detroit, USA, *IGRIP Users Manual*, 1995. IGRIP is a registered trademark of Deneb Robotics Inc.
- [59] R. Henriksson, *Scheduling Garbage Collection in Embedded Systems*. PhD thesis, Department of Computer Science, July 1998. LUTEDX/(TECS-1008)/1-164/(1998).
- [60] T. Lump, G. Gruhler, and W. Küchlin, "Virtual java devices; integration of fieldbus based systems in the internet," in *Proceedings of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON98)*, (New York, NY, USA), pp. 176–181, IEEE, 1998.

- [61] I. Entezarjou, "Internet- och javateknik för fältbussystem (in swedish; eng: Internet and java technology for fieldbus systems," tech. rep., Department of Computer Science, Lund University, November 1998. LUTEDX/(TECS-3081)/1-76/(1998).
- [62] F. S. Tamiosso, A. B. Raposo, L. P. Magalhães, and I. L. M. Ricarte, "Building interactive animations using vrml and java," in *Proceedings of X Brazilian Symposium on Computer Graphics and Image Processing*, (Los Alamitos, CA, USA), IEEE Computing Society, 1997.
- [63] "The java 2 platform." Sun Microsystems Inc. <http://www.javasoft.com/products/jdk/1.2>.
- [64] "The java tutorial." Sun Microsystems Inc. <http://java.sun.com/docs/books/tutorial/index.html>.
- [65] "The java3d api specification." Sun Microsystems Inc. <http://www.javasoft.com/products/java-media/3D/index.html>.
- [66] "The virtual reality modelling language." VRML Consortium, 1997. ISO/IEC 14772-1:1997.
- [67] "Cosmo player." PLATINUM Technology Inc., Visual Computing Group. <http://cosmosoftware.com>.
- [68] "The external authoring interface (eai)." PLATINUM Technology Inc. <http://www.cosmosoftware.com/products/player/developer/eai>.
- [69] T. Pattison, *Programming Distributed Applications with COM and Microsoft Visual Basic 6.0*. Microsoft Press, 1998.
- [70] C. Verbowski, "Integrating java and com," tech. rep., Microsoft Corporation, January 1999. http://www.microsoft.com/java/resource/java_com.htm.
- [71] "Specification of welding procedures according to en288." European Committee for Standardization. B-1050 Brussels.
- [72] B. Suhm, B. Myers, and A. Waibel, "Multimodal error correction for speech user interfaces," *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 1, pp. 60–98, 2001.
- [73] R. Rosenfeld, D. Olsen, and A. Rudnicky, "Universal speech interfaces," *Interactions*, pp. 34–44, November+December 2001.
- [74] P. R. Cohen, "The role of natural language in a multimodal interface," in *UIST92 Conference Proceedings*, (Monterey, California, USA), pp. 143–149, 1992.

- [75] M. A. Grasso, D. S. Ebert, and T. W. Finin, "The integrality of speech in multimodal interfaces," *ACM Transactions on Computer-Human Interaction*, vol. 5, no. 4, pp. 303–325, 1998.
- [76] J. J. Craig, *Introduction to Robotics*. Reading, Massachusetts, USA: Addison-Wesley Publishing Company, 1989.
- [77] ABB Flexible Automation, Västerås, Sweden, *RAPID Reference Manual*. 3HAC 7783-1.
- [78] Microsoft Speech Technologies. <http://www.microsoft.com/speech>.
- [79] J. N. Pires, *Industrial Robot Programming. Building Applications for the Factories of the Future*. New York: Springer, 2006.
- [80] <http://www.anoto.com>. Anoto homepage.
- [81] <http://www.logitech.com>. Logitech homepage.
- [82] "Api reference." Anoto SDK for PC applications (3.2), 2006. Anoto.
- [83] "Anoto sdk for pc applications," 2006. Anoto.
- [84] Microsoft, *Visual Basic Reference Guide*, 2005. Visual Studio .NET 2005 Documentation, MSDN.
- [85] <http://www.autodesk.com>. Autodesk homepage.
- [86] ABB Robotics, *ABB IRB1400 Users and Programming Manual*, 1995.
- [87] <http://robotics.dem.uc.pt/docs/video8>. Demonstration videos.
- [88] M. Haegele, "White paper on trends and challenges in industrial automation." November 2007.
- [89] JastAdd, "The JastAdd Compiler-Compiler System." <http://jastadd.cs.lth.se>, site accessed December 2007, 2007.
- [90] P. Buitelaar, "On the role of natural language processing in a data-driven approach to the ontology life-cycle." Keynote talk at TALN, Toulouse, France (<http://olp.dfki.de/ontoselect/>), June 2007.
- [91] D. L. McGuinness and F. van Harmelen, "OWL web ontology language." <http://www.w3.org/TR/owl-features/>, site accessed December 2007, 2007.
- [92] Protégé, "The Protégé ontology editor and knowledge acquisition system." <http://protege.stanford.edu>, site accessed December 2007, 2007.

- [93] SMErobot, “The European robot initiative for strengthening the competitiveness of SMEs in manufacturing.” 2007.
- [94] SIARAS, “Skill-based inspection and assembly for reconfigurable automation systems.” 2007.
- [95] RDF, “Resource description framework,” 2007. <http://www.w3.org/RDF/>, site accessed December 2007.
- [96] RDFS, “RDF schemas.” <http://www.w3.org/TR/rdf-schema/>, site accessed December 2007, 2007.
- [97] SMErobot, “Preliminary description of concepts for high-level programming methods,” Tech. Rep. Deliverable DR2.6, April 2007. Deliver.
- [98] XSLT, “XSL transformations (XSLT) version 2.0.” <http://www.w3.org/TR/xslt20/>, site accessed December 2007, 2007.
- [99] G. Antoniou and F. van Harmelen, *A semantic web primer*. The MIT Press, 2004.
- [100] SPARQL, “SPARQL protocol and RDF query language.” <http://www.w3.org/TR/rdf-sparql-query/>, January 2008.
- [101] SWRL, “SWRL: A semantic web rule language combining owl and ruleml.” <http://www.w3.org/Submission/SWRL/>, site accessed December 2007, 2007.
- [102] E. Friedman-Hill, “Jess, the rule engine for the Java platform.” <http://herzberg.ca.sandia.gov/>, site accessed December 2007, 2007.
- [103] J. Malec, A. Nilsson, K. Nilsson, and S. Nowaczyk, “Knowledge-Based Reconfiguration of Automation Systems,” in *Proceedings of IEEE CASE*, pp. 170–175, IEEE, September 2007.
- [104] J. Wielemaker, M. Hildebrand, and J. van Ossenbruggen, “Using Prolog as the fundament for applications on the semantic web,” in *Proceedings of the ICLP’07 Workshop on Applications of Logic Programming to the Web (ALPSWS-2007)*, (Porto, Portugal), September 2007.
- [105] VoiceXML, “Voice extensible markup language (VoiceXML) 2.1.” <http://www.w3.org/TR/voicexml21/>, site accessed December 2007, 2007.
- [106] O. Bersot, P.-O. El Guedj, C. Godéreaux, and P. Nugues, “A conversational agent to help navigation and collaboration in virtual worlds,” *Virtual Reality*, vol. 3, no. 1, pp. 71–82, 1998.

- [107] C. Godéreaux, P.-O. El Guedj, F. Revolva, and P. Nugues, "Ulysse: An interactive, spoken dialogue interface to navigate in virtual worlds. Lexical, syntactic, and semantic issues," in *Virtual Worlds on the Internet* (J. Vince and R. Earnshaw, eds.), ch. 4, pp. 53–70, 308–312, Los Alamitos, California: IEEE Computer Society Press, 1998.
- [108] E. Ammicht, E. Fosler-Lussier, and A. Potamianos, "Information seeking spoken dialogue systems part I: Semantics and pragmatics," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 532–549, 2007.
- [109] A. Potamianos, E. Fosler-Lussier, E. Ammicht, and M. Perakakis, "Information seeking spoken dialogue systems part II: Multimodal dialogue," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 550–566, 2007.
- [110] I. M. Delamer and J. L. M. Lastra, "Loosely-coupled automation systems using device-level soa," in *Proceedings of the 5th IEEE International Conference on Industrial Informatics*, pp. 743–748, 2007.
- [111] J. L. M. Lastra and I. M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Transactions on Industrial Informatics*, vol. 2, pp. 1–11, 2006.
- [112] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "Owl-s: Semantic markup for web services." W3C Member Submission, November 2004.
- [113] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. L. McGuinness, E. Sirin, and N. Srinivasan, "Bringing semantics to web services with owl-s," *World Wide Web*, vol. 10, no. 3, pp. 243–277, 2007.
- [114] S. Kona, A. Bansal, and G. Gupta, "Automatic composition of semanticweb services," *IEEE International Conference on Web Services (ICWS 2007)*, pp. 150–158, 2007.
- [115] Y. Chevalier, M. Mekki, and M. Rusinowitch, "Automatic composition of services with security policies," *2008 IEEE Congress on Services - Part I*, pp. 529–537, 2008.
- [116] <http://jacop.cs.lth.se>. JaCoP, Department of Computer Science, Lund University, Sweden.
- [117] C. Alexakos, A. Kalogeras, S. Likothanassis, J. Gialelis, and S. Koubias, "Workflow - coordinated integration of enterprise / industrial systems based on a semantic service - oriented architec-

- ture,” *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, vol. 2, pp. 273–279, 2005.
- [118] K. Charatsis, A. Kalogeras, M. Georgoudakis, and G. Papadopoulos, “Integration of semantic web services and ontologies into the industrial and building automation layer,” *EUROCON 2007 - The International Conference on “Computer as a Tool”*, pp. 478–483, 2007.
- [119] M. Paolucci, X. Liu, N. Srinivasa, K. Sycara, and P. Kogut, “Discovery of information sources across organizational boundaries,” *Services Computing, 2005 IEEE International Conference on*, vol. 1, pp. 95–102, 2005.
- [120] R. Vaculin and K. Sycara, “Towards automatic mediation of owl-s process models,” *IEEE International Conference on Web Services (ICWS 2007)*, pp. 1032–1039, 2007.
- [121] F. de Andrade, U. Schiel, and C. de Souza Baptista, “Constraint-based web services discovery and composition,” *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM’07)*, pp. 118–124, 2007.
- [122] T. Kirkham, D. Savio, H. Smit, R. Harrison, R. Monfared, and P. Phaithoonbuathong, “Soa middleware and automation: Services, applications and architectures,” *2008 6th IEEE International Conference on Industrial Informatics*, pp. 1419–1424, 2008.
- [123] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski, “Service-oriented architectures for embedded systems using devices profile for web services,” *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW’07)*, vol. 1, pp. 956–963, 2007.
- [124] A. Pohl, H. Krumm, F. Holland, I. Luck, and F.-J. Stewing, “Service-orientation and flexible service binding in distributed automation and control systems,” *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, pp. 1393–1398, 2008.
- [125] J. Lastra and J. Orozco, “Semantic extension for automation objects,” *2006 4th IEEE International Conference on Industrial Informatics*, pp. 892–897, 2006.
- [126] J. Mortimer, “Jaguar uses adaptive mig welding to join c-pillars to an aluminium roof section in a new sports car,” *Sensor Review*, vol. 26, pp. 272–276, 2006.