Extending an industrial robot controller with a fast open sensor interface — implementation and applications

A Blomdell[1], G Bolmsjö[3], T Brogårdh[4], P Cederberg[3], M Isaksson[4], R Johansson[1], M Haage[2], K Nilsson[2] M Olsson[3], T Olsson[1], A Robertsson[1,5], J J Wang[4]

[1]Dept. of Automatic Control, [2]Dept. of Computer Science, [3]Dept. of Mechanical Engineering, Lund Institute of Technology, Lund University, P.O.Box 118, SE-221 00 Lund, Sweden
[4]ABB Automation Technologies – Robotics and Manufacturing, Västerås, Sweden

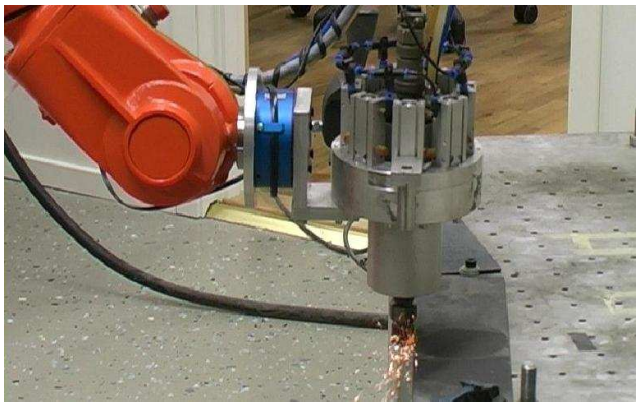

Figure 1: High bandwidth force controlled grinding, using an ABB Irb2400 industrial robot with the extended ABB S4CPlus control system described in this paper.

1. Introduction

Many promising robotics research results were obtained already during the late 1970s and early 80s. Some examples are Cartesian force control, advanced motion planning, etc. Now, 20 years and many research projects later, much of this has still not reached industrial usage. An important question to consider is how this situation can be improved for future deployment of needed 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. The principles used have been known for much longer, but deployment in products required affordable computing power, efficient engineering tools, customer needs for productivity/performance, improved end-user competence in utilization of performance features, etc.

However, applications that are considered non-standard 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 for those in small and medium enterprises (SMEs). Reasons include complex configuration, non-intuitive (for the shop floor) programming, and difficulties instructing robots to deal with variations in their environment. The latter aspect includes both task definitions and definition of motion control utilizing external sensors. The key word here is flexibility, and flexible motion control is particularly difficult since the user or system integrator need to influence the core real-time software functions that are critical for the performance and safe operation of the system. We have to find techniques that permit real-time motion controllers to be extended for new demanding application areas.

Open Control Most robot control systems of today support some type of user IOs 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 show up as longer duty cycles, whereas for some applications, like contact force control between the robot and the environment/workpiece, stability problems and severe performance degradation may result [1],[2],[3].

From a control perspective of robotics research, direct access to the driving torques of the manipulator and fast feedback is 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 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, but need to do so in a consistent way, supporting supervision and

[5]Corresponding author, Anders.Robertsson@control.lth.se

[6]The work in this paper was partly funded by the EU 5th Framework Growth Project GRDI-2000-25135: Affordable, flexible system for off-line automated fettling and finishing (AUTOFETT). The initial development of the external control platform and techniques was supported by VINNOVA, the Swedish Agency for Innovation Systems.

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, 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 paper, 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 instance [4].

1.1 Open issues

Developments and difficulties up to current state of the art raise fundamental questions that form the motivation of this paper:

- Industrial robot controllers today provide highly optimized model-based motion control, claimed 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, going from a text-book algorithm to a functional implementation required extensive engineering efforts. Today, we have engineering tools and code generation from specifications/descriptions/simulations of control principles. Comparing experimental work within the academic community with the industrial robot development, engineering tools such as Matlab, Maple, and the like, are quite similar, whereas code generation and deployment of controllers/components appear to be quite different. Deployment into a product requires substantially more verification, optimization, and tailoring to the system at hand. Then, the question is: Could commercial/optimized systems be structured to permit flexible extensions, even on a hard real-time level?

1.2 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-years lag between experiment and product, is high performance six degree-of-freedom control of the contact forces between the robot and its environment. As a part of the 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, and this will serve as our primary example.

What we will consider here are 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 name “sensor interface” may be a bit restrictive as it not only allows for feedback from external sensor data, but additionally for code and algorithms to be downloaded and dynamically linked into the robot control system (Fig. 2).

2. Considerations and design of system extensions

The architecture of the ABB S4CPlus control system and its extensions are shown in Fig. 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-level, Cartesian level, joint level, motor currents). Below we discuss the underlying design considerations and our implementation of the platform.

On a high level, there is already in the present ABB S4CPlus (and earlier) systems the possibility to read sensors via customer IO to influence the robot task as expressed in programs written in the ABB RAPID language. The RAPID program reading sensor information via the IO system can be referred to as a *pull protocol* which requires no external computing, but the sensor reading/handling must be expressed in the user program. There is today also the possibility to change programmed motion targets via Remote Procedure Calls (RPC) during robot motion, which can be referred to as a *push protocol*, requiring external computing but less RAPID programming, since the logic 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: The restriction that external sensors can be utilized on 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 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 IO data without self-description leads to less flexible application programs which require manual configuration, limiting development of high-level application program packages.

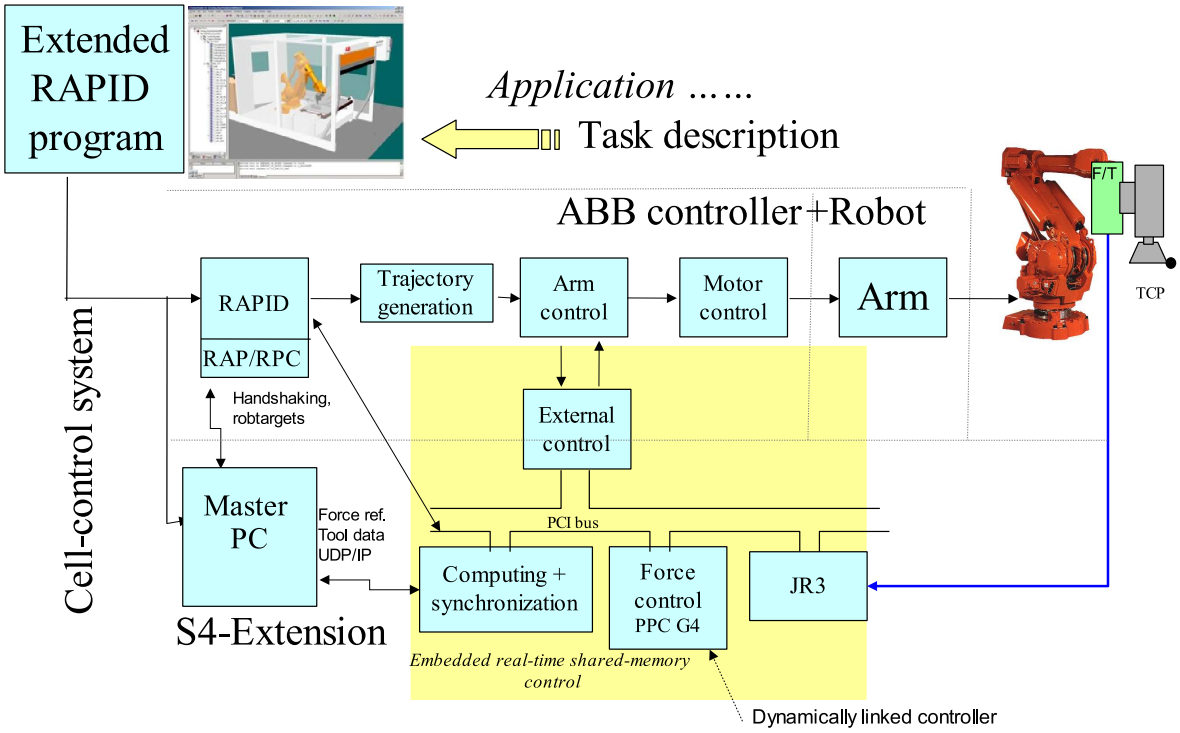

Figure 2: Extension of industrial robot controller with sensor interface and support for external computations and synchronization, using a Motorola PPC-G4 PrPMC-800 processor board mounted on a Alpha-Data PMC-to-PCI carrier board with a local PCI bus.

Thus, today we have high-level (user level) usage of low-level (primitive) sensors, but to overcome the two problems we also need low-level (motion control) usage of high-level (force, vision, etc.) sensors. Specifically, to start with, interfacing with force sensors should be supported. That is both a technically most demanding case and a most desired one from a customer point of view.

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

2.1 Hardware and IO protocol

The most promising hardware interfacing possibilities (from a cost and performance point of view) are shared memory access via the 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 IO 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 way 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 is 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 from Ethernet communication using raw scheduled Ethernet or UDP/IP show promising results [5]. The bandwidth is comparable to that of the PCI bus, standard network-order of data bytes simplifies interfacing, and with proper network/interrupt handling the latency can be very short, showing a great potential for future applications utilizing distributed sensors.

2.2 Safety and quality issues

Open systems require careful engineering not to exhibit unpredictable or even unsafe behavior when confronted with inexperienced users and extended with novel features at the customer site. One problem within development of open systems is the complexity in the systems engineering as such, where the following difficulties need to be addressed.

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

Firstly, 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. Secondly, third-party modules may hang the buses used by the control computers. Such a failure can be due to either faulty added hardware, or the bus interface of the added hardware being badly configured or accessed incorrectly.

In order 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. That can also be done by running the application with a virtual controller, and 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 since long been an important obstacle in real applications, and in cost-efficient production it is not as simple as saying that there should be redundant sensors, which is costly and increases the risk for 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 for 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 done by providing available header files. However, using the ordinary header files would expose too much of potentially highly classified motion control techniques. Therefore, there should be a neutral definition of (possibly) exposed variables, preferably based on information from text books or articles, and possibly suggested as an open standard.

Robot safety Even with hardware and software functioning as intended according to 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 is some cases, whereas other states (run-mode and brakes etc.) 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. Also that problem has been solved, however the difficult part is to combine safety with performance.

2.3 Performance

For industrial robots, control performance means productivity. Specific force control algorithms (inside the "Force controller" block in Figure 6) are outside the scope of this paper, but the imposed requirements on the open system deserve some attention.

Sampling and bandwidth considerations As an example, force control in a non-compliant environment typically requires fast sampling. The reason is that 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 described in Section 2, there are a number of levels in which external control actions can enter the system. Firstly, high-level feedback using the high level ABB RAPID language to modify the generated trajectories, gives a sampling time of h=0.1 s. The interface to the built-in arm servo control has a higher sampling frequency, h=4 ms. Finally, h=0.125 ms gives the maximum internal sampling frequency of 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 one degree of freedom 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 Fig. 3 we can see the results of the simulation using a rough surface of stiffness $k = 25$ N/mm, using a continuous time proportional controller, as well as discrete time control with the sampling times described above. The desired force F_r was 100 N. It can be seen that the response for $h = 4$ ms 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. In some applications however, such as force control in extremely stiff environments, or where high approach velocities are required, a higher sampling rate than 4 ms may be desired.

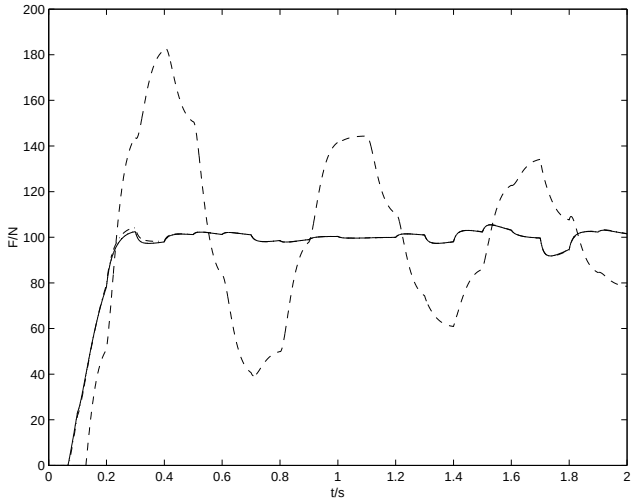

Figure 3: Contact force for continuous time design (solid), h=0.1 s (dashed), h=4 ms (dash-dotted) and h=0.125 ms (dotted).

```
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 4: Sample ExtRAPID program. The extended language constructs are located in RAPID comments and are modeled as XML tags in order to be easily modifiable.

2.4 User and system aspects

Open controllers need ways of expressing usage of sensors. The RAPID language of the S4CPlus controller support 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 of course 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 (Fig. 4). The processing of the XML comments is done in a new master PC module acting as a robot proxy, see Master PC in Fig. 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 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 low-level force control along the programmed path. This was accomplished by:

1. 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 Fig. 2) and fed to the Program Server of the S4C controller via the ABB RAP protocol.

2. 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 IO byte output later when the S4C servo actually performs the motion, that output—the sync signal from servo to master PC in Fig. 2—forming the lower byte of the integer value of a system-wide path coordinate.

3. 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 item 1, and since an external set-point in real time can influence the set-points to the Force Control Loop (Fig. 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 to what target the robot is commanded.

External motion control With language extensions and system connections in place, the implementation of the actual (to the S4C) external controller can be done. This is the force control in Fig. 2. To accomplish interrupt driven hard 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 IO hardware was carried out in our laboratory, as well as 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, 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 into the robot controller.

2.5 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, and models of the environment and sensors can be used for simulation. Tools such as Modelica and Dymola were used and shown to be very effective for modeling and simulation of many types of dynamical processes, including industrial robots [6, 7, 8, 9].

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 accuracy of programs created off-line especially since the introduction of technologies such as RRS. Extensive problems remain though, 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 taken despite changes in the state of the robot work-cell which 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 which allow new information and previous knowledge to be accumulated in a common format. High level re-planning of the robot task can then be automatically performed. Typical limitations of robot systems that are hard to handle on-line are 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.

Figure 5: Grinding with IRB6400 at Kranendonk, the Netherlands, using a compliant grinding tool developed at KU Leuven, Belgium, together with force feedback control. The video can be found at www.robot.lth.se.

3. Case study - force controlled deburring

The use of industrial robots for automated deburring, grinding and polishing is an interesting example of a process where external sensing capabilities are crucial. Accurate control of the contact forces can help increase the quality of the final product, as well as the flexibility in the deburring process. To handle the deviations from the nominal workpiece geometry that are inevitable consequences of the foundry process, some compliant behavior needs to be included in the system used for deburring. As an alternative to using a mechanically compliant tool or mounting of the workpiece, force control can be used to program a desired compliant behavior, and for maintaining a desired contact force during the deburring process.

3.1 Hybrid force/position controller

During the deburring task only the direction perpendicular to the surface of the workpiece is constrained, and a hybrid force/position control strategy is employed [10]. In this type of structure one or several degrees of freedom become force controlled, while ordinary position control is used in the other directions. Typically, the force controlled direction is perpendicular to the surface, while the motion tangent to the surface and the orientation is 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. There have also been extensions presented to the hybrid position/control approach, which take the robot dynamics into account [3].

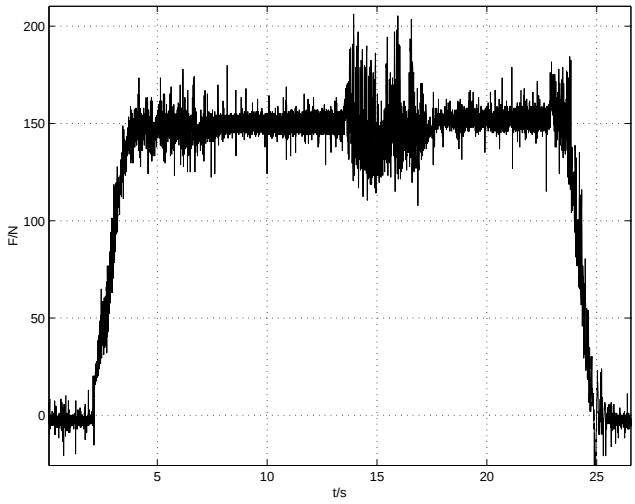

Figure 7: Contact force during grinding experiment with force reference 150 N. The disturbance at time $t \approx 13$ s is caused by a resonance in the workpiece.

3.2 Experimental results

Also included in the controller is a finite state machine responsible for switching between position- and force control, and for handling control during the transition and impact phases.

The grinding experiments were carried out on an ABB Irb6400 robot, see Fig. 5, at the company Kranendonk, the Netherlands, using a special grinding tool developed at KU Leuven, Belgium. The results from an experiment with reference $F_r = 150$ N is shown in Fig. 7. The disturbance that can be seen at time $t \approx 13$ s is caused by a resonance in the workpiece/fixture, which occurs when the grinding tool moves across a hole in the center of the workpiece.

4. Discussion

Robots are distinguished from other types of machines in terms of flexibility; that is, the ability to change their behavior through reprogramming, in order to be able to cope with new situations. From the initial research on programmable manipulation systems, recent research approaches typically fall into one of two categories; Autonomous robotics, with focus on handling unstructured environments but in large neglecting performance for industrial productivity, and industrial robots, with focus on motion performance in structured environments but neglecting most of the perception and navigation issues.

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. Then, we need to combine both theory and system technologies from various fields of robotics research. An important part is to package experimental results as useful components (for verification and reuse), and to find techniques that permit real-time motion controllers to be extended for new demanding applications, typically using external sensing to substantially improve flexibility.

Many robotics labs have reported activity on open control systems which fully satisfy the need for the above mentioned aspect on evaluation and implementation [11], [12].

The close cooperation and technology transfer between industry and academia has 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.

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). As an application and motivating example we report on the implementation of force controlled grinding and deburring within the AUTOFETT-project (EU Growth Programme).

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.

Experiences from the fully developed prototype and the industrial usage of it confirms 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

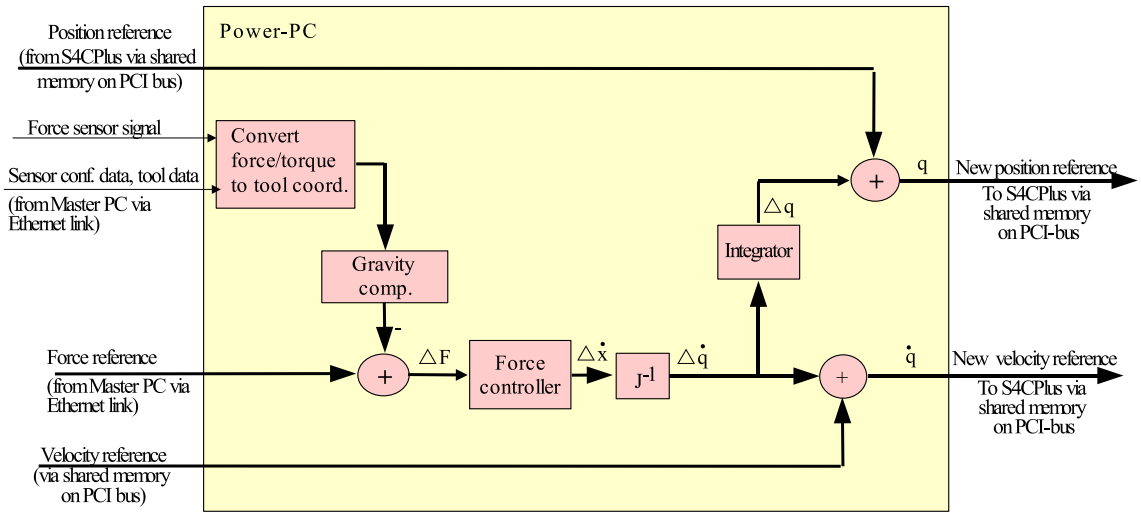

Figure 6: Force controller block structure. The force control algorithm is implemented inside the "Force controller" block.

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" which have been reported for academic research.

The mutual benefits of collaboration between academia and industry tend to be, we see, crucial for the future development of flexible productive robots: with open systems external partners will be able to extend the system also on the motion control level, and the richness of applications, their dynamics radically increasing the need for more research. In this process, it is not only a matter of technology transfer going from theory to practice, but a bi-directional flow of ideas and knowledge.

Acknowledgments

The authors are grateful to the friendly system-integration atmosphere and support at Kranendonk Production Systems in the Netherlands, and to the support within our organizations, including Håkan Brantmark and Peter Eriksson at ABB, and many others.

6. References

[1] B. Siciliano and L. Villani. *Robot Force Control.* Kluwer Academic Publishers, 1999.

[2] D. Gorinevsky, A. Formalsky, and A. Schneider. *Force Control of Robotic Systems.* CRC Press, 1997.

[3] T. Yoshikawa. "Force control of robot manipulators." In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 220–226, April 2000.

[4] H. Born and J. Bunsendal. "Programmable multi sensor interface for industrial applications." In *Proc. Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, pp. 189 – 194, 2001.

[5] A. Martinsson. "Scheduling of real-time traffic in a switched ethernet network." Technical Report Masters thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, March 2002.

[6] F. Calugi. "Observer-based adaptive control." Technical Report Masters thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, April 2002.

[7] Modelica, http://www.modelica.org.

[8] S. E. Mattsson and H. Elmqvist. "An overview of the modeling language Modelica." In Juslin, Ed., *Proceedings of the Eurosim'98 Simulation Congress*, pp. 182–186, Helsinki, Finland, April 1998. The Federation of European Simulation Societies.

[9] Dymola, http://www.dynasim.se.

[10] M. H. Raibert and J. J. Craig. "Hybrid position/force control of manipulators." *ASME Journal of Dynamic Systems, Measurement and Control*, **103:2**, pp. 126–133, 1981.

[11] K. Nilsson and R. Johansson. "Integrated architecture for industrial robot programming and control." *J. Robotics and Autonomous Systems*, **29**, pp. 205–226, 1999.

[12] H. Bruyninckx. "Open robot control software: the OROCOS project." In *Proc. Int. Conf. on Robotics and Automation*, vol. 3, pp. 2523–2528, 2001.