Introduction

Modern robot control systems used in industry at present 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.

The main objective in robot programming is to complete a desired set of tasks. This is accomplished by describing a sequence of robot motions and activities and teaching the robot these motions and activities. The teaching of the motion paths and activities can be divided into two groups: online and off-line programming. In online programming, the robot is involved at several stages during the programming process; in off-line programming, the robot does not participate in the program design, but uses programs that are ready-made or predesigned (Nilsson, 1996; Nilsson and Johansson, 1999).

Online programming of robots – also known as teach-in or show-and-teach – involves manual moves of the robot to each desired position. The resulting “program” is then a sequence of vectors (joint coordinates) plus activation signals from external equipment used to command the robot to perform the same motion at a later stage. Alternatively, the off-line programming is a language-oriented programming technique, similar to conventional computer programming, which enables the programs to be written in high-level or low-level language – sometimes with simulation and 3D graphical support (Nilsson, 1996; Nilsson and Johansson, 1999).

Task-level programming

In task-level programming, the desired goals of the task are commanded directly instead of specifying
every movement of the robot in detail. In this method, instructions can be added in the application program at a higher level in comparison with an explicit robot programming language. This scheme requires the system to have the ability to perform many planning tasks automatically.

Although the approach is a natural outgrowth of ongoing research and development in CAD/CAM and in robotics, task-level programming is still an open issue due to the difficulty in developing the language.

Task planning is an important part of this method and can be divided into three phases: modeling, task specification and robot program synthesis. Although these phases are not computed independently, they make the task of programming easier.

World modelling
The world model for a task must contain the following information:

- geometric description of all objects and robots in the task environment;
- physical description of all objects, for example mass and inertia;
- kinematic description of all linkages; and
description of robot characteristics, for example joint limits, acceleration bounds, and sensor capabilities.

In addition to such information, the position of all objects and linkages must be included in the world model. The uncertainty associated with each of the positions must also be specified. The main component of the world model is the geometric description of objects. The principle source of geometric models comes from the CAD systems. Throughout the execution of a task, information in a world model remains the same while the kinematic description of linkages changes. Hence, in the process of robotic operations, new linkages are created and old linkages destroyed.

During the planning of the robotic operation, a description of the work-cell characteristics – e.g. geometrical properties, mass and inertia of parts – must be included in the world model. Apart from these geometric, kinematic, and physical descriptions, a robot’s sensing capacities such as touch, force and vision, must also be considered.

Much of the complexity in a world model arises from the initial modeling of the robot. During each new task, information about the geometric, kinematic and physical models of the objects must be provided. In task-level programming, it assumes that all these items of information are developed as part of the design of the objects. If the assumption is not valid, the modeling effort required for a task-level specification using current modeling methods might dwarf the effort needed to generate a robot-level program to carry out the task.

Increased flexibility in industrial robot systems
Industrial robots have been used in flexible manufacturing systems since their introduction in the early 1970s. The possibility to handle a variety of products and better handle changes in production volume is the main motivation for flexible production system and industrial robots. However, there are still conflicts between current means of automation and the aims to flexibility in manufacturing. The use of robots for small batch or even one-off production is limited. Critical factors are the possibility to efficiently specify tasks without the need to heavily rely on well-calibrated world models and the possibility to assign “autonomous sub-tasks” relating the needed level of programming accuracy vs time.

The transition time has been reduced through the use of robot simulation systems (Cederberg et al., 2002; Olsson, 2002). In this way, the robot task can be simulated in a virtual model of the work cell when still only a digital prototype of the workpiece exists. The risk of technical failure for a transition can be reduced. The robot task description can even be transferred to the robot system to reduce the very time-consuming robot programming phase – i.e. task-level programming often being used synonymously with off-line programming.

A task-level programming system may work well as long as the world is regarded as static in the sense of the object placement and their geometrical shape. As such conditions are rarely fulfilled in the physical world, this dilemma is usually solved by applying complex clamping devices and unnecessarily high accuracy of the pre-manufactured workpieces. From an economical point of view, however, that is neither an efficient nor an appropriate method for single-piece or small-series production.

Feedback-supported task-level programming
The environments of robots are dynamic and must be observed by perceptional equipment. For adaptation of task realizations to the environment, the robot control system must have the ability to support and react to the observed information. An event-based robot control system with these advantages is to be described, the event-based control system operating from a model description of the world and the task. In the world model, all objects significant for the task in the robot work
cell are represented and generated during a visual task-oriented programming session. The task realization is managed by the control system in small segments or executable events. An executable event is fired and realized when its preconditions are fulfilled. Changes in the work cell are detected by sensors and the information is used to update the world model upon which all decisions on reactive events are based. Sensor information has influence both on planning and on control of motion and application processes. The main goal of this approach is to create the ability to autonomous management of task realization in a flexible environment (Nilsson and Johansson, 1999; Olsson, 2002).

Another focus is a task-oriented robot programming method and a discussion of the associated control system. The purpose of creating a new programming environment is to provide re-usability, maintainability and reliability to the robot program code, all key factors in efficient programming. The whole system is focused on objects corresponding to physical objects and processes in the environment. In the task-oriented programming system, tasks are described as states of objects and their dependencies. The assisting control system is event driven and operates with the objects as base for realization of events.

### Problem formulation

By using sensors during task execution, adaptation of the robot task is possible and the deviations of the world can be handled. We are not aiming for a completely autonomous industrial robot system, which would require an autonomous task-planner, but rather a system with good support for the robot user, who will specify the task-plan by using autonomous sub-tasks. In order to take intelligent decisions on how to handle detected deviations, the goal was to create a system where the execution of a robot task seamlessly moved from the simulation environment to the physical world. The simulation system should have the tools for the user to create and test a model that mimics the real system as closely as possible, the same model being used during the real-world execution.

In a series of feasibility studies, we showed that the feedback control should be integrated already at the task planning in order to provide desirable adaptation properties.

### Methods and experimental set-up

The following are the main parts in the system architecture with the interconnections shown in Figure 1.

- ABB Irb2400, ABB Irb6400, ABB S4Cplus;
- Task-level programming software Envision™/IGRIP;
- Force sensor JR³ (http://www.jr³.com), laser seam scanner, sensor interface and sensor feedback;
- Open robot control system (Nilsson, 1996; Nilsson and Johansson, 1999);
- Applications: arc welding, grinding, assembly; and
- Video camera Sony DFW-V300 640×480 pixels including firewire network connection.

#### Geometry model accessible during operation

A system was developed, where a commercial simulation engine was used both to contain the geometry model of the robot workcell and to execute the robot tasks. The state of the robot was continuously extracted and transmitted to an open robot controller responsible for executing the movements. Sensor information was propagated back to the model, assuring a valid up-to-date model to be available (Figure 2).

First, the system was used during experiments with a welding robot (Figure 3). The robot was equipped with a laser seam-tracker in order to record the position and orientation of the workpiece and some other technology data specific for the weld seam (gap, seam angles, etc.). The model was continuously calibrated with this information to detect and avoid typical problems during sensor controlled welding such as collisions, kinematic singularities and exceeding of joint limits.

#### Online management of robot limitations

Methods were developed to detect limitations during online execution of tasks and how they could be avoided. Algorithms to detect and handle exceeded joint-limits and kinematic singularities were developed. Experimental runs were performed to verify the functionality.

#### Force control

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 or for maintaining a desired contact force during the deburring process.

Among relevant force control strategies, it is relevant to consider the hybrid force/position control (Raibert and Craig, 1981) and impedance (or admittance control) (Dohring and Newman, 2002; Hogan, 1985; Siciliano and Villani, 1999).
During the deburring task only the direction perpendicular to the surface of the workpiece was constrained, and a hybrid force/position control strategy was employed (Raibert and Craig, 1981).

In this type of structure one or several degrees of freedom were force controlled, while ordinary position control was maintained in the other directions. Typically, the force controlled direction was perpendicular to the surface, while the motion tangent to the surface and the orientation was controlled using position control. The directions, which should be force controlled are selected using a diagonal selection matrix, which was 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 (Yoshikawa, 2000).

**Autofett**

Within the framework of the research program **Autofett**, the grinding experiments were carried out on an ABB Irb6400 robot at the company Kranendonk Production Systems BV, the Netherlands ([http://www.kranendonk.com](http://www.kranendonk.com)), using a special grinding tool developed at KU Leuven, Belgium.

**Virtual sensor development and robotic simulation**

In order to produce robust programs for robots that make use of sensors, there is a need for fast prototyping with integration of sensors as virtual...
sensors to task-level programming applications during simulation. To this purpose, a virtual sensor system was developed and a software model of a real sensor with its characteristics was designed, using geometrical and/or process specific data from a computerized model of a real work cell.

To achieve reliable simulation results, accurate dynamic models for the controlled robot system were important. The virtual sensor function was compared and validated with the real sensor through comparable measurements in set-ups. The experimental results with a virtual laser scanner behavior were measured and compared to its real counterpart.

A sensor interface for task-level programming

A sensor interface handling both virtual and real laser sensors was developed. Acting as a master, a task-level programming application continuously sent trajectories in joint coordinates to a slave robot reacting in real-time (Figure 4). The laser sensor was attached to the slave robot and sensor data were sent back to the master. The sensor interface, which handled messages between the two robots, was hiding sensor-specific issues by encapsulation. For that reason, the master was unaware of whether a virtual or real robot is used as slave. When the system ran in virtual mode, the laser scanner control unit was simulated as a subsystem to the sensor interface obtaining input from a virtual slave. In acquisition of sensor data – virtual or real – robot programs using sensor data did not have to be altered when transferred from the virtual world to the real robot cell (Cederberg et al., 1999, 2002).

Robot task specification

All models in the task specification contained information about the robotic environment and the objects being manipulated. The main difference in the models was the position of the objects known as the model states. During the execution of a task, the task was specified to the task planner as a sequence of models of the world state at several steps.

Task program synthesis

The synthesis of a robot program is an important phase in the task planning. It consists of three major steps: assembly sequence planning, motion planning and plan checking. The output of the synthesis phase is a program in robot-level language, which is useful for repetitious operations without the need for re-planning. This program consisted of assembly unit commands such as grasp commands, several levels of motion specifications, sensor commands and error tests.

Motion synthesis is a difficult task as it requires operating under uncertainty in the positions of the objects and of the robot. These uncertainties can be modeled and computed. In some cases, the uncertainty may be too large to allow any planned motions to be carried out. Hence, non-contact sensors must be added to send information to the task planner, which then decides the correct measurements to be made.

System implementation

The implementation of the experimental platform consisted of a dedicated network solution connecting the robot simulation program IGRIP/Envision, the external sensors such as for instance the laser seam tracker, and the open robot controller (ORC) at the Robotics lab of the Department of Automatic Control.
(Nilsson and Johansson, 1999; Robertsson et al., 2000). The communication with the simulation program was handled via the low-level teleoperation interface in “real-time mode”, allowing generated trajectories to be sent to the robot system and, what is crucial for the concept presented here, the objects in “the virtual world” – i.e. in the “world model” – to be updated from sensor data and robot measurements with approximately 15-25 Hz (Figures 2 and 3). The necessary routines and code needed for the interface to the robot system were dynamically linked and updated during run-time – i.e. the robot control system did actually need to be fully configured at the start of the task execution. Direct access to lower hierarchical level (allowing control loops with faster bandwidths) was also available, for instance, to introduce fast weaving during welding.

Results

Force control
Using combined feedback from vision and force sensors, we achieved a surface following accuracy independent of the calibration accuracy of the workpiece. The tool was guided over the surface of the workpiece by visual feedback, while a desired contact force was maintained by the force controller, the resulting contact forces when the tool moved over a corner in the workpiece are shown in Figures 5 and 6. The selection of what workspace directions of force and position to be controlled at a given time, and the parameters of the controller, were changed online based on the sensor data available.

As an alternative to using a mechanically compliant tool or mounting of the workpiece, force control was used to program a desired compliant behavior, or for maintaining a desired contact force during the deburring process. A situation, where programmable force directions via active force or compliance control have large benefits to a flexible tool is shown in Figure 7. During the circular trajectory (grinding around the inside of the tube) the robot does not need to re-orient the tools.

During the deburring task, only the direction perpendicular to the surface of the workpiece was constrained, and a hybrid force/position control strategy was employed (Raibert and Craig, 1981). In this type of structure, one or several degrees of freedom were force controlled, while ordinary position control was used in the other directions. Typically, the force controlled direction was perpendicular to the surface, while the motion tangent to the surface and the orientation was controlled using position control. The directions, which should be force controlled were selected using a diagonal selection matrix, which was set as a part of the high-level task specification. Also, extensions to the hybrid position/control approach, taking the robot dynamics into account, were presented (Yoshikawa, 2000).

Autofett

The grinding experiments were carried out on an ABB Irb6400 robot at the company Kranendonk Production Systems BV, the Netherlands, using a special grinding tool developed at KU Leuven, Belgium. The results from an experiment with \( F_r = 150 \text{ N} \) is shown in Figure 8. The disturbance at time \( t \approx 15 \text{s} \) seen in Figure 8 was caused by a resonance in the workpiece/fixture, which occurred when the grinding tool moved across a defect in the workpiece surface.

As seen in Figure 8, the contact force was maintained at a constant value, while the grinding tool was moved across the surface. From an analysis of the noise spectrum, it was verified that the noise signal was periodic, probably caused by the fast rotation of the grinding stone. The rotation caused vibrations in the robot, workpiece and fixture.

Open control architecture

The open control architecture effectively integrated workspace sensors (laser seam scanner,
force sensor, and camera) with the regular robot sensor devices and other exteroceptors. As a result, sensor fusion among the real-time data from the work-space sensors and from the regular robot sensor devices was made possible. In turn, re-calibration of the world model and modification of the task execution accordingly were effectively accomplished. Thus, task execution was accomplished for poorly defined initial positions and orientations of the work objects as well as for work objects disturbed in mid-term task execution (Figure 2).

The possibility to close feedback loops in a consistent way, bridging the large difference in time scales and hierarchical orders of the system, was an important means to the performance and flexibility of the platform.

Using sensor data, the world model was continuously re-calibrated to detect and avoid typical problems during sensor controlled welding such as collisions, kinematic singularities and exceeded joint limits (Figure 1). Obstacle avoidance proved impossible without a valid world model accessible during execution.

**Figure 6** Measured forces $F_x$ and $F_y$ during vision-guided contact force experiment (Figure 5).

**Figure 7** Situation where programmable force directions via active force or compliance control have large benefits to a flexible tool.

**Note:** The reference is given as constant normal force to the surface. At time $t = 22$ the grinding tool reaches the corner and we can see the corresponding change in the measured forces.

**Note:** During the circular trajectory (grinding around the inside of the tube) the robot does not need to re-orient the tools.
**Online management of robot limitations**

Although greatly improving performance, the result of a robotic task execution was not always a success even if the model of the world was correct and continuously updated by sensors, and the process knowledge was complete and accurate. As a worst case, if unforeseeable problems appear and limitations of the robot system prevent execution, the total robot task had to be abandoned.

**Force-control automotive power train assembly**

Assembling automotive power trains is a traditionally manual work, performed by skilled operators with years of experience. This is because gears and other critical components of the clutches, torque converters and so on, have to be aligned with very high precision. Such operations, however, take their toll on human labor. Tedious and fatiguing, they can lead to repetitive-motion stress injury, lower product quality and a drop in efficiency. A robot able to perform at the same level as a human operator would obviously make the prospect of automation in this application area very attractive.

If a position-controlled robot tries to align a pair of gears and its control program does not have precise information about the gear-tooth positions, the robot's sole option is iterative trial and error, repeated until the relative tooth positions are found. Any attempt by the robot to make the gears as long as they are misaligned will cause one gear to press hard against the other, generating unacceptably high contact forces. Even, if the teeth were chamfered to facilitate mating, misalignment would still produce large side forces as the robot struggles to move the gear along the pre-programmed path toward the centerline of insertion. More than likely, the gears would even jam unless some means of mechanical compliance is provided.

A case in point is the assembly of Ford's F/N torque converter case, which weighs about 25 kg. Inside this case, there is a double splined gear-set into which a pump gear has to be inserted. The seal of the pump gear is critical and great care has to be taken to ensure that it is not damaged in any way during the insertion. An internal splined shaft has to be fitted at the same time, thus complicating the assembly.

The tests carried out with advanced force control in the automotive industry have convincingly demonstrated its ability to improve the cycle time and agility in different assembly applications. In one application involving the insertion of a forward clutch, a work cell with an IRB4400 robot averaged 5.7 s for the insertion with a reaction force of less than 100 N. In another, F/N torque converters were assembled in an average time of 6.98 s with the contact force limited to 200 N (Figure 9). Here it is worth noting that, in addition to the part itself weighing about 25 kg, the allowed positional tolerance was ±2 mm.

![Figure 8](image1) Contact force during grinding experiment with feedback-controlled force reference 150 N

![Figure 9](image2) Assembly time histogram for torque converter (IRB6400, Vacuum Gripper)
Discussion

In our feasibility studies, we studied performance and flexibility in the task-level programming and task execution of:

- virtual force control;
- force representation in CAD task planning;
- Simulink including generation of code; and
- robotic force control application in grinding, deburring and assembly.

In many robot applications, such as arc-welding, spot-welding, spray painting with circular brush, milling and drilling, the process only requires five degrees of freedom. As a majority of industrial robots have six degrees of freedom, the extra degree of freedom leaves room for choice among an infinite number of kinematic robot configurations fulfilling the given task, e.g. avoiding kinematic singularities. Although the idea is not new, the solution has been rejected in the past with the motivation that it is impossible to use this extra degree of freedom without jeopardizing system safety. Problems like collisions or tangled cables in the case of arc welding could arise. In our experimental system, continuous collision check using updated geometry models was accomplished, thus increasing reliability without compromising production quality.

However, applications that considered non-standard at present 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, in particular for those in small and medium enterprises. 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 is flexibility, and flexible motion control is particularly difficult since the user or system integrator would 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.

Performance

The restriction that external sensors can be utilized on the conventional unmodified system 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. Therefore, force control as needed within several application areas such as foundry is currently quite difficult to accomplish.

Flexibility

The use of port-based IO data without self-description leads to less flexible application programs which require manual configuration and limiting development of high-level application program packages. IO-ports are global resources, so the use of them in software resembles the use of global variables. Additionally, the performance problem also implies that the robot is less flexible since new unforeseen application demands often cannot be solved at the customer site.

Distributed sensor feedback over networks

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 fieldbus interfaces and communication introduce delays and limit performance as compared to the shared memory interface. Whereas, control performance means productivity for industrial robots, specific force control algorithms are outside the scope of this paper, but the imposed requirements on the open system deserve some attention (Figure 10).

Sampling and bandwidth considerations

Note that system delays and the sample period matter a great deal in the context of force control. In the architecture of the ABB S4CPlus system, there are a number of levels in which external control actions can enter the system corresponding to different effective sample periods \( h \). 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 the control theory that feedback from a sampled signal decreases the stability margin, thereby decreasing the robustness to varying operating conditions. In order to achieve high-performing force control, it is necessary to provide a force-sensor interface with high-bandwidth properties.

Virtual sensor development and robotic simulation

Considering the number of robots used in industrial automation, the use of advanced sensors such as vision, laser scanners or force/tactile sensors is still small. However, the current trend towards fast product changes along with
customization and optimized design using new materials and manufacturing processes, put greater demand on manufacturing operations with respect to control performance and the resulting productivity and quality. One such example is arc welding, where products based on new materials decrease the overall dimension (plate thicknesses) and increases the general need for keeping tight tolerances during welding.

To be able to produce robust programs for robots that make use of sensors, there is a need for integrating sensors as virtual sensors to task-level programming applications during simulation. Through this, a tool will exist that makes it possible to understand how a robot system operates under sensor guidance without setting up a real robot system for that purpose. In this context, a virtual sensor is a software model of a real sensor with similar characteristics, using geometrical and/or process specific data from a computerized model of a real work cell. To achieve reliable simulation results, accurate dynamic models for the controlled robot system may be of great importance.

We described the steps taken to develop a sensor model as a virtual sensor implemented in a task-level programming package that can be used for simulation of a robot with sensors. The virtual sensor function was compared and validated with the real sensor through comparable measurements in setups. In this case, a specific sensor was described, but the method is general and practically any type of sensors should be possible to model and integrate to a task-level programming system, such as force sensors and 2D vision. The experimental results, where a virtual laser scanner behavior was measured and compared to its real counterpart successfully validated its performance.

Robot task specification
Several methods were developed to specify the model states, such as the spatial relationships, which describe the relative position of the objects. In this method, all positional dimensions were given based on a planar face or a cylindrical center...
line. However, this method had several limitations. Another alternative to task specification was through the description of a sequence of operations. Therefore, during the operations the users could describe the required tasks or operations instead of building a model of an object in the desired position.

During real-time task execution, the actual position of the object differed from those in the model. These differences were caused by part variations, robot positioning errors and modeling errors. Thus, to make the robotic program useful, it was always important to consider the worst scenario in design. As such systematic programs are difficult to write and have a longer execution time, the task planner had to use his/her experience to decide which motions and strategies are efficient and robust, thus challenging deterministic behavior. If the uncertainties were too large to guarantee success, then additional sensors or fixtures had to be used to limit this uncertainty. Owing to this reason, the estimation of uncertainties played a major role in task specification.

**Task program synthesis**

Task planning assumes that the difference between the actual state of the world and the world model is only within some known bounds. As the work objects may be outside the range of the estimated uncertainty, however, such assertions are not always true, the objects being of the incorrect type or even not present. Hence, synthesized programs written should be able to detect and correct any failure so as to prevent the robot and other parts of the environment from being damaged. In conclusion, it is necessary to add sensory tests in robot programming for error detection.

In practice, standard task-level programming systems available are useless for the simulation of sensor controlled system. An ideal simulator utilizes the task-level programming method to generate a program, which can be tested for validity. This type of simulator incorporates robot motions and sensory information into a dynamic world model. This in turn provides the planner with information about events in the world. By using sensors during task execution, adaptation of the robot task is possible and the deviations of the world can be handled. For reasons of increased system complexity, industrial reluctance to sensor technology is perceptible in cases, where it is impossible to predict the exact outcome of a run. They have to be extended with functionality for sensor models and robustness analysis. Such analysis includes the probability to enter kinematic singularities, collisions or joint limits.

**System implementation**

The system architecture also allows to make an “interactive off-line programming” and evaluation of proposed solutions by introducing simulation models of the sensors and the robot system (Cederberg et al., 1999). For conceptual evaluations, only a fairly crude dynamic model of the (controlled) robot system seems to be needed, while for accurate matching with the real robot system more refined models will be needed. The robot controller simulation (RCS) modules provided by many of the large robot system manufacturers suggest one such solution, and may be directly incorporated in simulation software (Olsson et al., 1999).

**Exception management and hybrid systems**

During execution of a task, failure detection should cause activation of exception management whether it be foreseeable feedback modifications close to a kinematic singularity, activation of some new, reactive task or a fatal event prompting task abortion. As to their nature, the resulting control systems may be characterized as hybrid control systems, which currently receive considerable attention (Johansson and Rantzer, 2003).

The system architecture presented consists of the interconnection of a commercial robot simulation program and an open robot control system with a task dispatcher configuration via a dedicated network according to Figure 3. The time scales and real-time requirements differ a lot in the different parts of the system. For the case of using a shared network, the simulation environment with virtual sensors offers a good tool for performance evaluation of the task-execution with respect to time delays introduced in the network connections.

A different approach was taken in the Esterel/ORCCAD effort (Berry, 1998; Borrelly et al., 1998; Simon et al., 1993, 1997), where formalized reactive systems were implemented. The analysis of hybrid systems with the combination of continuous time and discrete-event systems was supported. The ORCCAD objective was to generate, simulate, execute and verify automata including properties of liveness, conflict detection and conflict resolution.

In this context, we should also mention the Free/Open Software project supported by the European Community on Open Robot Control Systems, OROCOS, the aim of which is not on the system architecture, but on providing software components and component frameworks (OROCOS, www.orocos.org).
Conclusions

This paper describes the design and implementation of a platform for fast external sensor integration into an industrial robot control system (ABB S4CPlus). The sensor interface accomplished has interesting features due to the combination of the following.

1. A shared memory interface to the built-in motion control, enabling fast interaction with external sensors. Sensors can be handled in both hard real-time (using kernel-space modules) and soft real-time (using set-point input from master, with the external controller acting as a feed-forward).

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 the 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 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 is currently used for force controlled grinding and force controlled assembly.) 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 online 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. 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 and needs radically increases the requirements for more research.

Performing high level control with a world model updated in real-time from sensors in a work-cell, real or virtual, using a sensor interface yields several benefits.

1. High level control can be moved outside the actual work-cell. More effective coordination in a specific work-cell and among different work-cells is therefore possible.

2. Robot programs can be tested in a virtual world and later in a real work-cell without rewriting of code, a property which cuts development time and increases robustness.

3. Online tests, collision tests, out of joint limit tests and others can be performed in advance or in real-time as soon as sufficient information regarding the real world becomes available to the virtual world model.

4. Since the virtual model is updated continuously, knowledge of the process can be accumulated. This property may be very valuable for task execution improvements using the concept of iterative learning control (ILC) (Arimoto et al., 1984; Norrlof, 2000; Robertsson et al., 2002).

The sensor interface and the methodology to apply it in task-level programming applications support modeling and analysis of robot stations that include sensors to produce robust and validated robot programs.

References


Further reading