On-line singularity avoidance, collision detection and task level programming utilizing a world model

Per Cederberg, Magnus Olsson, Gunnar Bolmsjö
Div. of Robotics, Lund University, P.O. Box 118, S-221 00 Lund, SWEDEN
{pcederberg, molsson, gbolmsjo}@robotics.lu.se

Abstract

Knowledge of the environment is a critical component in robotics automation. The use of advanced sensors may yield the competitive edge in robotized small batch and one-off production systems. However, sensors increase the system complexity. To avoid, for instance, singularities and collisions, decisions have to be taken at a comprehensive level, during run-time or even before a process is started. This collection of system knowledge has been given a name - the world model.

In this paper, a world model is utilized to address two important decisions during an arc-welding process: 1) singularity prediction and avoidance and 2) collision detection with a simple re-planning of trajectory. Furthermore, the system design allows a mix of real- and simulated components using a task level approach.

Key words: robotics, sensor, simulation, real-time, world model, CAR, Computer Aided Robotics, arc welding, virtual sensor, simulated sensor, manufacturing

1 Introduction

The increase of the number of robots in automation has been and still is crucial to stay competitive in modern production. Despite the intrinsic usefulness of a six axis robot, normal utilization does not reach beyond reiteration of preprogrammed trajectories. While static robot programs may be sufficient to high volume manufacturers, they are not adequate in one-off or small batch production. The emerging interest in open control systems [Nilsson, 1996; Sperling and Lutz, 1997] and the introduction of advanced sensors may give future robotized small batch manufacturing the platform needed to avoid high costs of advanced clamping, expensive workpiece preparation and touch-up. Research in this area include simulation and animation [Chen and Trivedi, 1994], agile manufacturing [Quinn and others, 1996], adaption of sensors to a nominal model [Behrens and Roos, 1995], sensors applied to autonomous robots [Kugelmann, 1994], simulation of vision systems [López de Meneses and Michel, 1998] and space robotics [Brunner and others, 1993; 1994; 1999; Hirzinger and others, 2001].

Even with open control systems, there are several pitfalls on sensor usability. To resolve run-time related process issues, we promote a world model. Our world model includes a suitable design and runtime environment adequate for 1) creation of a program involving sensor guided robots, 2) simulation of the plausible variations and effects of sensor-robot interaction with the work-cell and 3) execution and supervision of the very same program in a real work-cell on the shop floor.

In traditional off-line programming of industrial robots, information about the work-cell is lost when the program is created for the target robot system. Since we are using advanced sensors, our application needs continuous information about work-cell changes during the process.

Advanced CAR (Computer Aided Robotics) environments such as IGRIP, RobCAD and RobotStudio are reasonably useful off-line development tools. Their monolitic design, lack of parallelism and internal dependencies, however, make them inappropriate as on-line runtime engines. The lack of parallel execution means that the execution speed sometimes is non-deterministic even for slow processes such as arc-welding. Furthermore, dependent on the extent of openness of the target robot controller, motion execution conflicts may occur between the CAR application and the robot controller. This is the case with the ABB S4C controller, a fairly closed controller despite its Robot Application Protocol (RAP) interface.

Thus, while the various off-line CAR environments have their niche, it is difficult to find appropriate environments for development, simulation and execution of advanced sensor guided robots. The authors have devel-

oped a library, RLib, which is capable of handling high level motion control, communication event handling, etc. in our research projects.

2 Objectives

In this paper, a world model is created and utilized to address two important decisions during an arc-welding process that includes advanced sensors: 1) singularity prediction and avoidance and 2) collision detection with a simple trajectory re-planning. The system design allows a mix of real and simulated components. Furthermore, it operates on task level and uses an unmodified ABB 2400.16 robot. The overall objective is to be able to use robots in small batch and one of production systems.

3 Related work

Implications of world models have been discussed in the mobile robot community for quite some time now. The focus in mobile robotics is navigation and mapping, including sensor interpretation, collision avoidance, localization and path planning. The goal is to create robots that operates in natural and unknown environments. [Kortenkamp and others, 1998] contains thirteen case studies of successful mobile robot systems and represents current state of the art from leading universities and research laboratories.

The world of industrial robotics is different in the sense that the focus is not on *how* to navigate. Instead, the objective is to perform a task under hardware, process and quality constraints. Examples of constraints in this paper is; the industrial robot controller is more or less a closed system; we would like to avoid expensive clamping and workpiece preparation; motion close to singular positions affects the process quality. The research is a work in progress. [Cederberg *et al.*, 2002b; 2002a; Olsson, 2002] expound our views in earlier efforts.

4 Experimental setup

Our example, an arc-welding application, uses only standard components, the M-Spot Laser Scanner with CSR4000 control hardware, two Unibrain Fire-i-400 cameras each equipped with a 3.5 mm lens, and an ABB 2400/16 robot with an unmodified S4C system. Several features are handled by our system components; remote compilation and loading of a specially designed RAPID program to the S4C controller and on-line trajectory generation. The S4C controller only executes pre-loaded RAPID programs. Therefore, the program has been given a generic design which demands continuous updated joint values to execute. Figure 1 gives a logical view of system components and Figure 2 shows the actual setup.

4.1 Application

The application connects with sockets to the tracker and ultrasound interfaces, feeder and IGRIP. The information time rate needed for a M-Spot sensor guided robot during arc-welding is in the area of 40-60 ms (20Hz) which makes local host socket communication a suitable choice.

4.2 Feeder

To accomplish on-line trajectory generation, a "feeder" component routes the joint values from the application to the generic RAPID program executing on the S4C controller. This system component utilizes the ABB specific RAP. From the application's point of view, the requirement is to deliver joint values whenever requested by the feeder. The feeder is also responsible for reading the actual pose of the robot.

4.3 Tracker

A tracker component has been developed to allow applications to interact with the M-Spot control unit, without having awareness of communication issues such as internals of data packets, error handling procedures etc. When the tracker component is used for simulation, a simulator emulates the CSR4000 actions. A simulated tracker, i.e. the tracker running a simulator subcomponent, may work in conjunction with a real or simulated robot. Again, this does not affect the application which is truly unaware of whether the tracker connects to the simulator or to the CSR4000 control unit. If a simulated robot is used, a simulated camera emulating the M-Spot, can be applied. A simulated robot could be an ordinary robot from a robot library in IGRIP, RobCAD or any other application providing visualization and an API that enables the user to create a simulated camera for interaction with the work-cell objects. By having the camera in IGRIP emulate laser rays sweeping over the camera's field of view and working distance, an array of distances can be collected. The sweep distance array can then be sent to the "slave" which is another tracker sub component.

The slave sub component performs image processing on the sweep distance array using similar algorithms as the CSR4000 control unit. The process result consists of a set of break points, i.e. information describing the geometry of the chosen joint type. If no such joint can be discovered, an error is returned. The process result is used by the "master" which is the tracker sub component closest to the application. Whenever the application wishes to receive information from the tracker, the request is handled by the master sub component, which in turn talks to either the CSR4000 or the simulator, without knowledge of which. The decision whether to simulate or not is taken by the user when the tracker

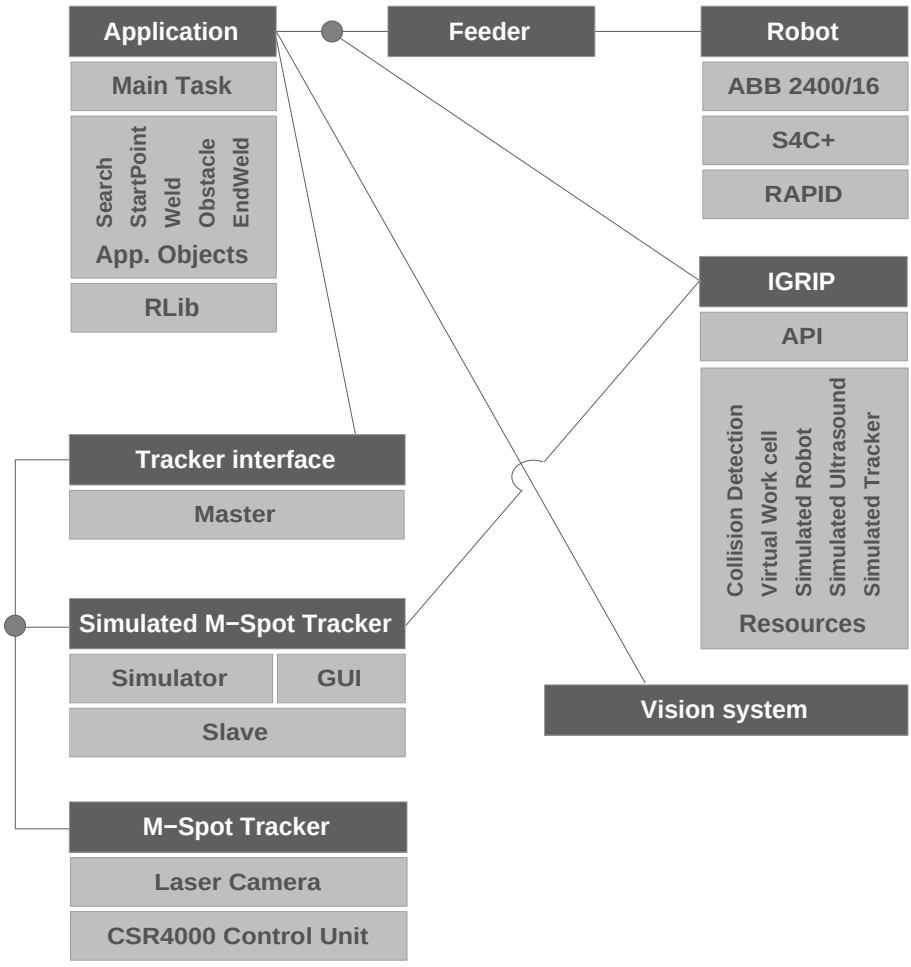

Figure 1: Logical view of system components in the experimental setup. TCP/IP is used for communication between components. The dots mark "either-or" relationships. In the application-IGRIP relation, however, there is always a connection to the virtual work-cell components in IGRIP and the simulated robot. When the system uses a physical robot, a simulated physical robot follows the real robot motion.

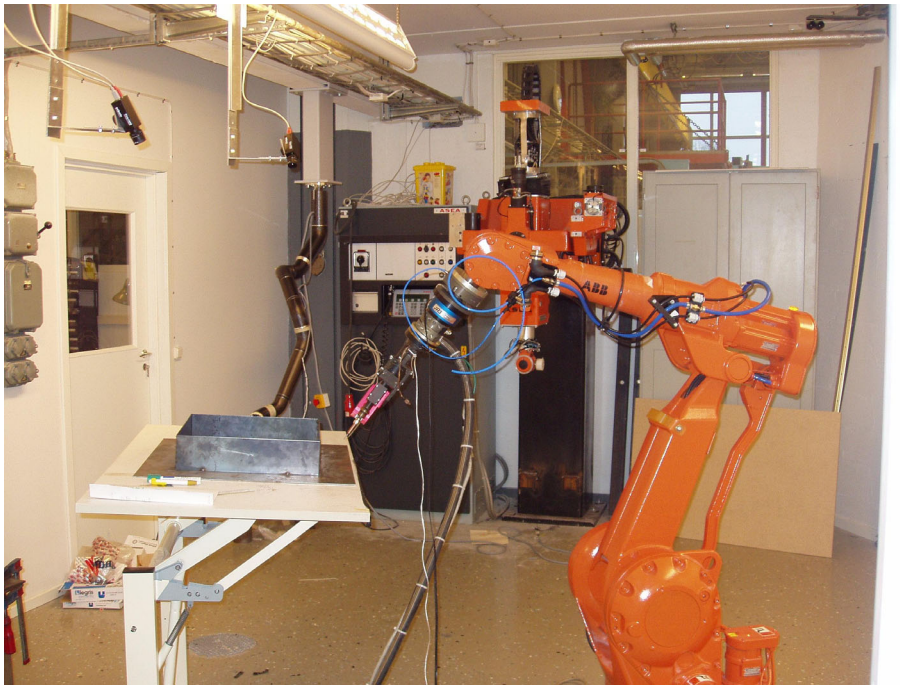

Figure 2: Acutal setup (without reinforcement). An ABB 2400/16 robot with an unmodified S4C system (outside view), the M-Spot Laser Scanner attached 50 mm ahead of the weld torch with its CSR4000 control hardware (outside view) and two Unibrain Fire-i-400 cameras, each equipped with a 3.5 mm lens.

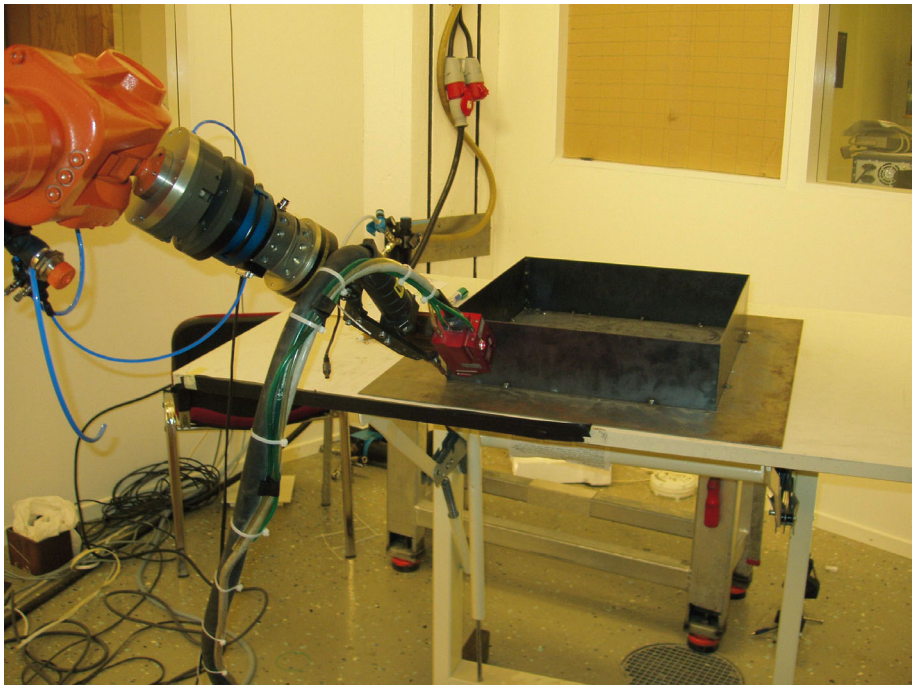

Figure 3: The system during the weld phase.

starts up. When running the tracker in simulated mode, a tracker GUI can be used to monitor the joint and the calculated breakpoints.

4.4 Stereo cameras

Two cameras are applied above the workpiece to yield an estimated position and orientation. The cameras also measure the distance to the obstacle.

4.5 IGRIP and its resources

Delmia's IGRIP, our CAR application, provides resources for creating the nominal virtual work-cell and the simulated physical workcell, collision detection in the simulated work-cell and display of virtual and simulated objects. In simulation mode, IGRIP also displays the "physical" robot motion and distance measurements utilized by simulated sensors. The resources in IGRIP are reach by its proprietary APIs.

4.6 Application objects

The application consists of a number of independent objects, where instances of some of the objects can be utilized (and re-utilized) in virtually any order. The objects encapsulates the logic in different phases of the application; search phase, start point phase, weld phase, obstacle avoidance phase, end weld phase etc. For all application objects, the user defines object specific behavior such as speed, weld current and other process dependent data. Several instances of an object may be utilized in one application.

Search object

The search object is utilized to find the weld start point. When the start point is found, it creates a start point path and a weld path. The search continued after the start point is found to be able to measure the weld workpiece orientation and thus, the weld direction. This second phase in the search has been given a name: The extended search. The extended search distance is usually equal to the distance between the weld torch and the laser scanner, and the search information is saved as a number of tag points in the weld path, each containing a pose. A search path is a user defined number of sweeps in the weld direction. After the extended search, the virtual workpiece is calibrated to the actual pose of the (possibly simulated) workpiece measured in the search.

Start point object

This object consists of the start point task which in turn executes along the start point path that was created by the search object. Usually, the start point path only consists of a single start tag point.

Weld object

The weld object continuously reads data from the laser scanner and creates tag points during the weld. In other words, new poses are appended to the weld path during the weld. Instances of this object can be utilized anywhere between the start point object and the end weld object.

Obstacle object

The obstacle object responsibility is to safely guide the robot in the vicinity of the obstacle. The robot will follow a predefined obstacle path. The obstacle is the path's base coordinate system, and depends on the shape of the obstacle. During this phase, data from the laser tracker is disregarded.

End weld object

The end weld object is similar to the weld object, except that a lost weld message from the seam tracker is interpreted as end-of-seam instead of being interpreted as an error.

4.7 Application object interaction mechanisms

While it is comfortable to split the weld task into logically defined discrete sub tasks using the above mentioned objects, since robot motion is continuous, there is a need to know when one object should hand over responsibility to another. Instances of, for example, a weld object can occur after a search object or after an obstacle object and therefore the object initialization prerequisites must also be known. The strategy utilized in this implementation is to refer to common motion data by calls to objects owned by the motion control library, RLib (briefly described below). Application specific data, such as references to the robot, tracker and ultrasound sensor, is handled through a shared process object which is passed to every application object during initialization.

4.8 RLib, the high level motion control library

All components, except for the tracker GUI, are built upon objects from RLib, a light-weight library for high level simulation and high level control of advanced sensor guided robots in soft real time. It has not been the authors primary interest to develop RLib, but it was necessary to be able to handle the interaction with the unmodified S4C controller. RLib is written in portable ANSI C in an object oriented style and handles model building including frame dependencies, trajectory creation, singularity avoidance and motion. The library is based on POSIX p-threads and includes API:s to handle threads and synchronization between objects. RLib-objects are for example tag-points, paths, tasks and robots.

While RLib handles the kinematic relationships between these objects, IGRIP is utilized for visualization,

collision detection, export of the nominal kinematic relationships and for simulating the M-Spot camera. In this context, the term nominal refers to known knowledge of the work-cell's frame dependencies before any input from sensors ("how we think the world looks like").

RLib also contains an API which can be utilized to import kinematic relations from a work-cell in IGRIP, RobCad, RobotStudio or any other application. Provided that the application allows traversal of its internal data, a RLib database can be created and read into a runtime database during initialization of the application. The application objects can then modify the runtime database through an API. Other features in RLib, besides motion handling, are;

- device independence. Handles all devices that can be described with the Denavit-Hartenberg notation;
- synchronization and thread handling routines;
- kinematic expression of any RLib object in any other RLib object;
- conversion functions between homogeneous coordinates and other representations;
- event handling, notification and subscription routines;
- communication, printing and serialization routines.
- safety routines and debugging help.

4.9 The world model

In the experimental setup described, the world model is split between the application, RLib and IGRIP. RLib is responsible for kinematic relationships handling and for generating robot motion, while IGRIP provides graphical feedback, measurements needed by simulated sensors and collision tests. The two environments are kept in synch during run-time by the application and therefore every change in RLib immediately affects the graphical model in IGRIP. The opposite also holds; a detected collision in IGRIP propagates instantly to the application which in turn asks RLib to halt execution.

5 Experimental work

The experiment objective is to show a capability to handle process-related events during runtime in a system where real and simulated objects (robots, sensors, workpieces) are transparently interchangeable. The on-line events are

1. calibration of world model objects based on sensor input;
2. trajectory creation based on calibrated objects;
3. singularity detection and algorithm based avoidance performed in the calibrated world model;
4. collision detection of simulated objects in the calibrated world model;
5. and simple re-planning of trajectory to fulfill the weld task.

The workpiece was essentially a four sided box. The objective of the first part of the experiment was to find the workpiece actual position and orientation. The size of the workpiece was known in advance. The calibrated cameras yielded a rough estimate the workpiece position and orientation as well as an accurate relative position of the reinforcement placement. The application adjusted the position of the virtual workpiece and reinforcement to coincide with camera values.

A search path covering the vicinity of the nominal (expected) placement was followed. Figure 4 shows a simplified search. The search area occupies less than 10×10 cm. It depends on the accuracy of camera results for the given workpiece and light conditions. The search could contain several sweeps from left to right. Since the path is expressed in workpiece coordinates, the path followed the workpiece calibrated pose. When the joint eventually was found, the start point was calculated based on the break points from the tracker. If the tracker was unable to find the joint during the search, the process was aborted and the robot returned to its home position. The start point was then saved as the first tag point in the newly created trajectory - the weld path, see Figures 3 and 6.

The camera was mounted 50 mm ahead of the tool tip and the search continued this distance and accumulated information about the joint. This extended search phase, see Figure 4, yielded joint-local information and the actual orientation of the workpiece. The extended search resulted in an update of the nominal world model to the workpiece real position and orientation. All objects referring to the workpiece, also automatically received their correct pose.

Before the robot was to follow the weld path, a preliminary singularity check was conducted. The check was performed as a simulated weld, i.e. a virtual robot was moved along the weld path and the check was done for each interpolated pose. The utilized algorithm is based on the fact that arc-welding is a five axis process since the weld quality is not affected by a rotation about the weld wire. Unfortunately, to yield a correct result, the M-Spot camera needed to be perpendicular to the weld direction. This led to a decision whether to abort the process due to a close-to-singular pose, or to accept a possible lower weld quality caused by loss of tracker break points. This is a case-by-case decision depending on weld quality requirements. In this particular process, it was decided to prioritize singularity avoidance.

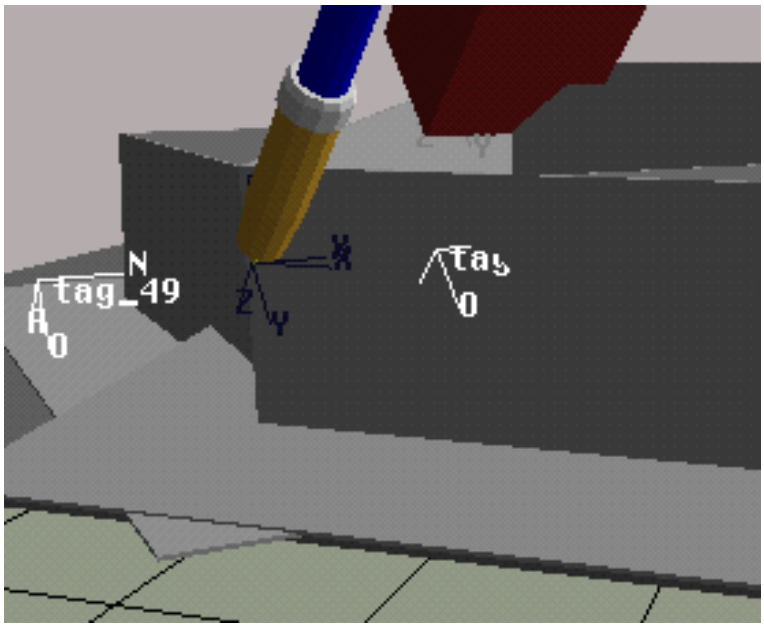

Figure 4: The search phase. The semi-hidden box in the background represents the nominal placement of the work piece. In the foreground, the actual pose of the box is simulated. The search path follows the nominal weld direction. In this snapshot taken from a simulation, a simplified search path with only one sweep is shown for clearness. It is possible to define several sweeps which then will show up in IGRIP as a forrest of tag points. The simulated M-Spot tracker is mounted on the weld torch, 50 mm ahead of the weld wire.

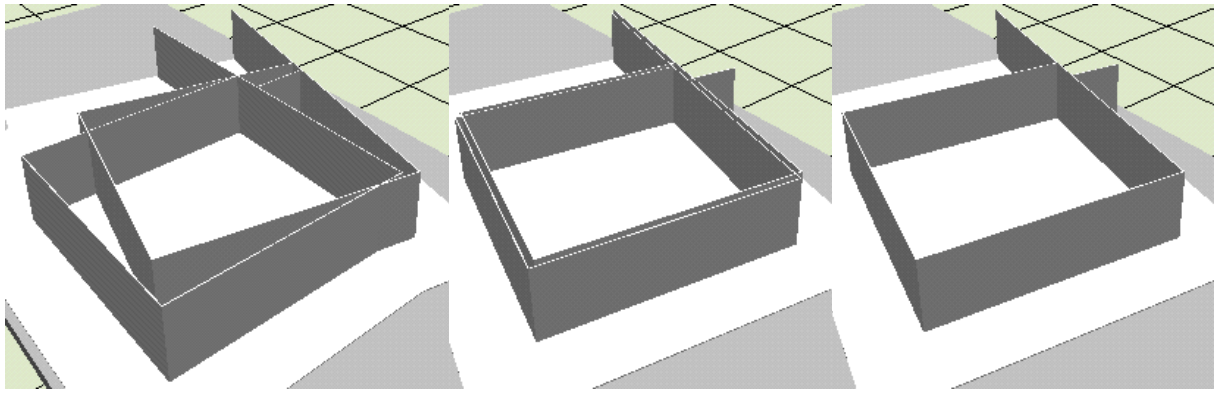

Figure 5: *(Left)* The workpiece before calibration from the vision system. The cameras yield a good estimate of the workpiece position and orientation. In the vertical direction, the error is less than five millimeters and in the horizontal direction less than 25 mm. The reinforcement's pose is accurately given as an offset from the weld start point. *(Middle)* After the camera calibration but before search with the laser tracker. *(Right)* After searching with the tracker, the position and orientation error of the workpiece is sufficiently small to create a good quality weld

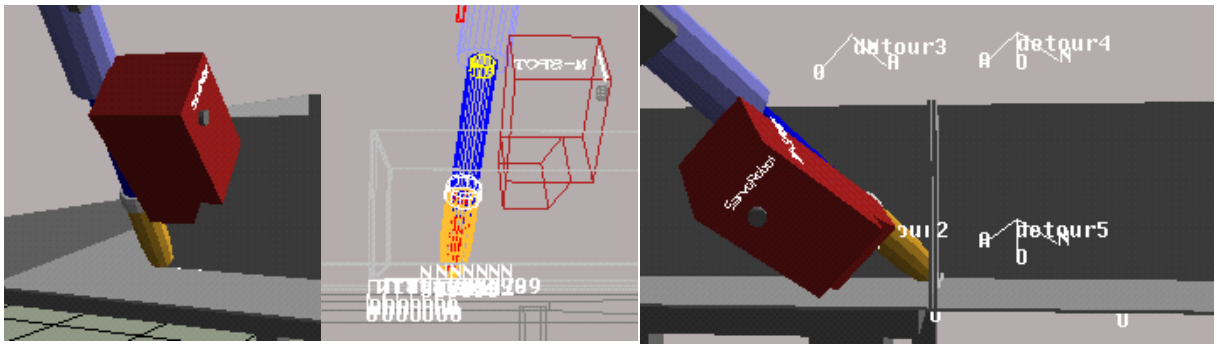

Figure 6: *(Left)* The start phase. *(Middle)* The weld phase. After the search phase the virtual workpiece is calibrated by the application. The simulated tracker sends arrays of distances to the tracker slave which calculates break points describing the joint profile. The application uses the break points to calculate tag points which it appends to the weld path. These tag points are then used by RLib to calculate joint values that create the trajectory for the (simulated) robot. The tag points already created at the bottom of the middle picture represents the start point and other poses found during the extended search. These tag points become first poses in the weld path. *(Right)* The obstacle phase. The path around the obstacle is expressed in virtual obstacle coordinates which in turn are expressed in the virtual workpiece. Since the virtual workpiece at this point is calibrated, the obstacle path (marked as detour in the figure) will create a valid trajectory around the reinforcement after it is calibrated. By using a path expressed in obstacle coordinates, it is easy to exchange obstacle without affecting application code.

The actual pose of the workpiece may induce a collision between the robot and objects in the work-cell and therefore collision detection was also performed in parallel with the singularity check during the simulated weld. Optimized collision avoidance and trajectory re-planning which are specific research areas that were not particularly studied in this paper. The obstacle was a tack welded reinforcement with a known size and orientation perpendicular to the weld seam but in an unknown absolute position measured in the weld direction. This type of reinforcements support the structure during welding only and are then removed. Having an understanding of what kind of obstacle to expect and an approximate position of where to expect them is relevant in industrial robotics and should not be interpreted as a model restriction.

The precise location of the obstacle was given after the beginning calibration. The vision system yielded the information needed - the distance from the weld start point to the reinforcement.

The reinforcements were designed to let the weld continue uninterrupted below it. To be able to approach the reinforcement and to create a continuous weld, the M-Spot camera was rotated 120 degrees, away from the workpiece and the weld torch was tilted a 45 degree angle from the weld direction, see 6. When the weld on the left side of the was done, the weld torch was retracted in the opposite torch direction and followed a trajectory relative to the reinforcement. By using a world model, it is simple and desirable to create trajectories relative other objects, see Figure 6. This produces an association between the sub-task program, the trajectory and the real object, and encapsulates their dependencies. The association can then be reused to handle similar sub-tasks in different contexts.

Even though the trajectory is known in advance, it may not be collision free. If the path passes too close to a singular pose, the robot's configuration may then unexpectedly change with an unpleasant result. Therefore a singularity and collision check was necessary. If the check failed, the tag points had to be either moved or reoriented. The second weld path on the other side of the reinforcement required similar checks as the first, but now the start of the weld was critical with the weld torch tilted to reach the end point of the first weld.

6 Results

Based on a nominal world model and added knowledge from sensors during runtime from physical and simulated components, weld process decisions have been taken at a comprehensive level and singularities and an obstacle have been detected and avoided using trajectory re-planning.

The development of a runtime environment and a simulated sensor, a laser scanner, has made it possible to simulate the weld process before deployment, as well as executing the unmodified application on a standard ABB 2400/16 S4C system equipped with a M-Spot CSR4000 seam tracker and by using two Fire-i-400 Unibrain cameras equipped with 3.5 mm lenses.

The on-line motion control of the fairly closed S4C system was possible due to the development of a "feeder"

which routed joint values using the ABB Robot Application Protocol from the application to a generic RAPID program running on the S4and Leuze VRTU 430 ultrasound sensor system, and RLib, which handled motion and miscellaneous issues.

The application was based on reusable objects, each dedicated to a specific task. Thus, we can think of the task as beeing solved by having an operator choosing from pre-defined sub-tasks to which process specific data were added.

7 Discussion

The research addressed the problem posed by manufacturing systems that must deal with product changes on a frequent basis. The use of simulated sensors and the methodology developed to model sensor systems provides a good resemblance between reality and simulation. Such modeling allows the manufacturer to deal with the shorter product life cycles required by current and future customer demands. It permits sensor-based systems to be analyzed at an early stage.

The research responds to the demands for flexibility in the use of sensors in coping with various facets of production. However, applying sensors in industrial automation is not as simple as it seems at a first glance. Sensors add not only valuable information needed to perform a task, but also system complexity. Thus, the robustness of the system may be degraded as a result of poorly integrated subsystems.

It is the view of the authors that the sensor feed-back must include several layers in time space and information complexity and a holistic control perspective to meet goals set up in a task or process specification. Traditionally, sensors are used to feed back information to a low-level type of actuator or process control. Advanced application processes and a higher level of autonomy implies more complex relationships; observable parameters are not necessarily those that are controllable and the controllable parameters are not necessarily those that define the task. Hence, in complex industrial operations, there are mapping issues in both directions between not only observable parameters that are detected by sensors and controllable parameters, but also between the task specification described in terms of how to reach productivity and quality measures and how to control the process to obtain such goals. In most cases this is not a trivial problem, as many controllable parameters are contradictory. As a result, many such issues must be considered as optimization problems that, due to the complexity and incoming information from sensors in real-time, must be solved on a case by case basis.

The conceptual framework developed has the advantage of being a platform for both simulation and actual operation. It operates on the system level and cooperates with modern CAR applications but can easily be hosted by any program with a graphical world model. As the same components and protocols are used for both environments, dynamic effects originating from the internal system are taken into account. The system is generic to such an extent that it handles simulation and operation without any change to the application.

Future work should include development of several useful application components, for instance inner and outer corner objects. Furthermore, a GUI that let the operator define the task without creation of any code further emphasizes the benefits of this novel type of task level programming should be developed.

8 Conclusion

It can be concluded that performing high level control with a world model updated in real-time from sensors in a work-cell, real or simulated, using a sensor interface yields several advantages;

1. High level control can be moved outside the actual work-cell. More effective coordination in a specific work-cell and between different work-cells is therefore possible.
2. Robot programs can be tested in a simulated world and later in a real work-cell without modification of any system component. This cuts development time and increases robustness.
3. On-line tests, collision tests, out of joint limit tests 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 can effectively be used in error recovery during robot operations.

Acknowledgements

The authors wish to thank Dr. Jacek Malec, Dept. of Computer Science at Lund University, Sweden for advice on mobile robotics, and Dr. Stefan Adolfsson, LTH School of Engineering in Helsingborg, Sweden for supporting the project with stereo vision.

References

[Behrens and Roos, 1995] A. Behrens and E. Roos. A method to adapt off-line programmed manufacturing tasks to the real environment using high resolution sensor devices. In *28th International Symposium on Automotive Technology and Automation*, pages 121–129. ISATA, 1995.

[Brunner and others, 1993] B. Brunner et al. Multisensory shared autonomy and tele-sensor-programming – key issues in the space robot technology experiment rotex. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2123–2139. IEEE/RSJ, July 1993.

[Brunner and others, 1994] B. Brunner et al. Task directed programming of sensor based robots. In *International Conference on Intelligent Robots and Systems*, pages 1080–1087. IEEE/RSJ, September 1994.

[Brunner and others, 1999] B. Brunner et al. A universial task-level ground control and programming system for space robot applications. In *5th International Symposium in Artificial Intelligence, Robotics and Autonmation in Space*. SAIRAS, June 1999.

[Cederberg *et al.*, 2002a] Per Cederberg, Magnus Olsson, and Gunnar Bolmsjo. Remote control of a standard abb robot system in real time using the robot application protocol (rap). In *Proceedings of the International Symposium on Robotic, ISR2002*, Stockholm, October 2002. IFR. paper No. 113.

[Cederberg *et al.*, 2002b] Per Cederberg, Magnus Olsson, and Gunnar Bolmsjo. Virtual triangulation sensor development, behavior simulation and car integration applied to robotic arc-welding. *Journal of Intelligent and Robotic Systems*, 35(4):365–379, December 2002.

[Chen and Trivedi, 1994] C. Chen and M.M. Trivedi. Simulation and animation of sensor-driven robots. *IEEE Transactions on Robotics and Automation*, 10(5):684–704, 1994.

[Hirzinger and others, 2001] G. Hirzinger et al. Space robotics - towards advanced mechatronic components and powerful telerobotic systems. In *6th International Symposium on Artificial Intelligence*. SAIRAS, Juni 2001.

[Kortenkamp and others, 1998] D. Kortenkamp et al., editors. *Artificial intelligence and mobile robots: case studies of successful robot systems.* The MIT press, 1998.

[Kugelmann, 1994] D. Kugelmann. Autonomous robotic handling applying sensor systems and 3D simulation. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, pages 196–201. IEEE, 1994.

[López de Meneses and Michel, 1998] Y. López de Meneses and O. Michel. Vision sensors on the webots simulator. In *Virtual Worlds 98*, pages 264–273. LNAI, July 1998.

[Nilsson, 1996] K. Nilsson. *Industrial Robot Programming.* Dept. of Automatic Control, Lund University, Lund, Sweden, 1996. Ph.D. Thesis.

[Olsson, 2002] Magnus Olsson. *Simulation and execution of autonomous robot systems.* PhD thesis, Division of Robotics, Department of Mechanical Engineering, Lund University, Sweden, March 2002. CODEN: LUTMDN/(TMMV-1051)/1-100/2002, ISBN 91-628-5120-9.

[Quinn and others, 1996] R. Quinn et al. Design of an agile manufacturing workcell for light mechanical applications. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 858–863. IEEE, April 1996.

[Sperling and Lutz, 1997] Wolfgang Sperling and Peter Lutz. Enabling open control systems - an introduction to the OSACA system platform. In *Proceedings of Robotics and Manufacturing, ISRAM 97*, number 6. ASME, 1997.