Advanced Engineering Informatics 24 (2010) 329–339

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A skill-based approach towards hybrid assembly ☆

Frank Wallhoff [a,*,1], Jürgen Blume [a,1], Alexander Bannat [a,1], Wolfgang Rösel [b,1], Claus Lenz [c,1], Alois Knoll [c,1]

[a] Human–Machine Communication, Technische Universität München, Theresienstraße 90, 80333 München, Germany
[b] Institute for Machine Tools and Industrial Management, Technische Universität München, Boltzmannstraße 15, 85748 Garching, Germany
[c] Robotics and Embedded Systems, Technische Universität München, Boltzmannstraße 3, 85748 Garching, Germany

ARTICLE INFO

Article history:
Received 12 May 2010
Received in revised form 19 May 2010
Accepted 27 May 2010
Available online 26 June 2010

Keywords:
Adaptive control
Hybrid assembly
Instruction based learning
Multi-modal interaction
Worker surveillance

ABSTRACT

Efficient cooperation of humans and industrial robots is based on a common understanding of the task as well as the perception and understanding of the partner's action in the next step. In this article, a hybrid assembly station is presented, in which an industrial robot can learn new tasks from worker instructions. The learned task is performed by both the robot and the human worker together in a shared workspace. This workspace is monitored using multi-sensory perception for detecting persons as well as objects. The environmental data are processed within the collision avoidance module to provide safety for persons and equipment. The real-time capable software architecture and the orchestration of the involved modules using a knowledge-based system controller is presented. Finally, the functionality is demonstrated within an experimental cell in a real-world production scenario.

1. Introduction and related work

Flexibility and adaption to rapidly changing market demands is one of the highest design criteria for the working cells and production facilities of the future. Therefore, researchers focus on a novel upcoming production paradigm, i.e. the so-called hybrid assembly. In contrast to existing either fully automated or purely manual production sites, the goal inside the hybrid assembly cell is to allow a collaborative joint-action in a workspace shared by an industrial robot and a human worker.

Nowadays, for the handling of heavy work pieces in automated production environments concepts of human–robot cooperation are developed [1]. A sample task of such a hybrid system is to assemble heavy parts of an automotive rear axle. Other applications feature mobile platforms with a mounted industrial robot arm, with which, e.g. welding processes are performed [2].

In general, the cooperation between humans and industrial robots is emerging more and more in ongoing research fields, concerning human–machine interaction as well as industrial safety requirements, issues and regulatory decrees [3,4]. Humans and robots that need to co-operate efficiently must share a common workspace, certain knowledge about the task to solve, about each other, and about side constraints including the environment.

In such workplace and time sharing systems the worker and the robot are additionally able to jointly handle objects or perform an assembly task together. Therefore, robotic manufacturers are developing new *safe robots* to enable working besides each other. The newly integrated safety features can replace the need for the robotic system to suspend its work when humans are too close [5]. Not only research tries to bring together human and robot in the same workspace, but also industry sees the potential and the great benefit for human–robot teaming in industrial production. At the end of 2008 more than one million robots were in operation world-wide [6]. Therefore, a great potential market lies in the equipment of existing industrial robots with new safety measures for hybrid assembly applications.

Within the PowerMate system [7], the interaction between robot and human worker is supported via a force–torque-sensor. Thereby, the robot can be moved by the human operator measuring the push and pull forces captured with this device. The safety in this interaction is provided by reduced velocity of the robot's speed and the usage of a two-hand control switch on the worker's side. The underlying safety concept fulfills the requirements according to DIN ISO 954, category three. However, the robot is not allowed to move on its own within the shared work space, without the workers active approval – he has to activate the control switch.

A further approach can be seen in [8], where a small robotic manipulator itself is additionally equipped with capacitive sensors

☆ This document is a collaborative effort from TUM-CoTeSys – Cognition for Technical Systems.

* Corresponding author. Tel.: +49 89 289 28552; fax: +49 89 289 28535.
E-mail address: wallhoff@tum.de (F. Wallhoff).

[1] All authors contributed equally.

1474-0346/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2010.05.013

to detect contact with the body of the robot. Here, the robot reduces speed on approach of the worker towards its body.

In the SMErobot project [9], the needs and requirements for the use of industrial robots in small and medium enterprises has been studied extensively. Basing on this requirement analysis, new robots and teach-in methods (e.g. Programming by Demonstration, voice, gestures or graphics) were developed and tested together with the end-users. By applying these new methods the human can program the robot more easily, e.g. by direct haptic teach-in of trajectories. Afterwards, the robot can perform this task accordingly.

As a further example approach, collaborative robots or *cobots* (invented by Edward Colgate [10,11]) can be named. These robots are mechanical devices that provide guidance through the use of servomotors, while a human operator provides motive power.

Further approaches [12–14] to ease the programming of interactive systems are applying Augmented Reality, Programming by Demonstration and learning from observation.

For more efficiency in the collaboration, it is necessary for the robot to move autonomously within the shared working environment to fulfill its current task with regard to the worker's safety. Therefore, the robot needs to recognize its environment. In this article, a method of enabling interaction with a standardized industrial robot is described, using additional multi-sensory devices within an experimental cell for hybrid assembly stations (Section 2). Potential field methodology is combined with a constraint least-square optimization to block the robot from moving into prohibited areas. The approach presented in this article is to re-plan the robots motion, using the information about the perceived environment (Section 3.3) to avoid obstacles and try to fulfill the current task without an emergency shutdown.

A further important aspect within the interaction with the robot is programming new tasks. According to Nakaoka et al. [15], robot programming by learning from observation consists of three different areas: task-, skill- and body-recognition. One major challenge is to make the system operable in a non-constrained environment and to generalize the problem [16]. Therefore, a hybrid assembly cell was set-up in a real factory environment (non-laboratory conditions), in a so-called Cognitive Factory [17].

Within this article, a method of instruction based learning is applied, using skills from each connected module, to generate a new hybrid assembly task. Both robotic and human skills can be applied within the task, which is then stored and executed using a *knowledge-based system controller* (Section 4). It is to be emphasized, that not only the teach-in process is performed under shared workspace conditions, but also the execution of the task during the actual production. Hence, to make the interaction within the joint assembly more intuitive, multi-modal communication channels are offered to the worker featuring speech, gaze and haptics (Section 3.1).

To accomplish this challenge, a sophisticated network of involved modules and hardware components needs to be set up. In this approach the orchestration of the modules is organized by the knowledge-based system controller. To satisfy the heterogeneous communication needs of the involved modules, a mixture of the three architectural paradigms (Service Oriented, Component Oriented and Distributed Object Architecture) presented in [18] is applied.

The interplay of the presented methods and components is explained within an application scenario.

Following the introduction, this article is structured as follows: in Section 2 an overview of the experimental cell layout is given. Hereafter, the software architecture and selected relevant modules are introduced in Section 3. Thereafter, the knowledge-based system controller is explained in more detail (Section 4), followed by the description of the application scenario in Section 5. The paper closes with the conclusions and an outlook.

2. Experimental cell layout

To have a realistic test bed for the developed concepts, an experimental hybrid assembly cell was set up. This cell, which can be interpreted as an agent as proposed by Shen [19], is embedded in the overall scenario of the *Cognitive Factory* [17] in between fully automated and manual assembly and thereby constituting a multi-agent environment [20]. In the selected cell layout, a human and a robot are arranged opposite to each other around a jointly used workbench sharing the same workspace with a size of 0.7 m × 1.0 m. Fig. 1 depicts the hybrid assembly cell with a conveyor belt on the right hand side. This conveyor belt connects the aforementioned assembly agents with each other to allow a fast and flexible exchange of required or processed parts. Furthermore, it provides all attached stations access to a central storage.

The installed industrial robot (Mitsubishi RV-6SL) is assisting the human worker. This robot manipulator can carry objects with a maximum weight of six kilograms, features six degrees of freedom and has a manipulation area that lies within a radius of 0.902 m around its body. The tool center point is extended with a force–torque-sensor with six degrees of freedom and a tool-change unit. Two tool change stations with three tool ports each are placed in the workspace, where different kinds of end effectors can be stored and exchanged. Currently the following tools are available to be changed autonomously by the robot, depending on the next processing step:

- two finger parallel gripper
- electronic drill
- camera unit for automatic observations
- gluer
- pallet gripper

This set of manipulators provide the robot with the capabilities to complete different tasks, e.g. drilling or grasping of objects. Depending on the requested robot operation, the gripper manager (see Section 3.5) initiates the changing towards the appropriate gripper autonomously.

As already mentioned in Section 1, the robot has to perceive its environment and interact with the worker. Therefore, several kinds of sensing devices (input) and a projection unit (output) are mounted above the workbench (see Fig. 2) to survey and inform the human worker. Two top-down view cameras are installed

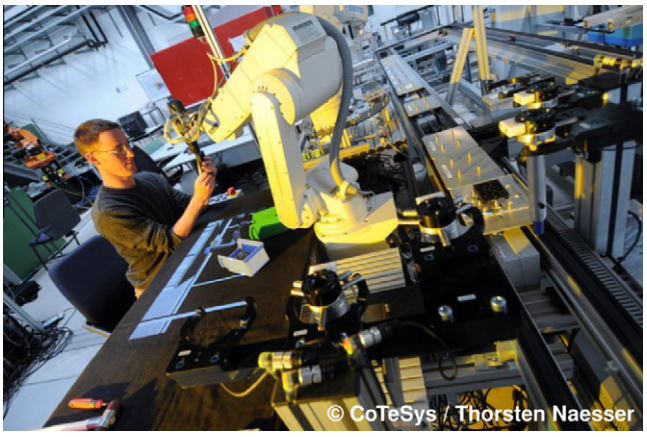

Fig. 1. Hybrid assembly station: human worker drilling together with industrial robot.

Fig. 2. Sensing devices and a projection unit: two video-cameras, a Photonic Mixer Device camera (PMD) (back, left to right) and a video projector (front).

above the workbench providing a global view of the shared workspace. These perspectives are processed to monitor the actions on the workbench and locate objects on the desk. Additionally, a depth sensing camera (Photonic Mixer Device – PMD) is used redundantly to provide information about obstacles or actions in the workspace using, comprising the camera data with depth information. The sensory input for the shared workspace surveillance unit is constituted by the PMD, weight enabled mats and a laser scanner in front of the desk. To interact more natural with the system, a desktop microphone is used to capture utterances of the worker. Furthermore, gaze information about the worker can additionally be captured using head-worn tracking glasses (e.g. to control sensitive interaction fields).

For interacting with the worker, a table projector is used to overlay needed information into his field of view directly onto the surface of the workbench. This includes assembly instructions or sensitive interaction fields.

To embed this station into the overall production concept of the Cognitive Factory, the hybrid assembly station is equipped with a Radio-Frequency Identification (RFID) system. The idea behind the usage of this innovative technology, which is applied in many different fields [21], is to have a product centric manufacturing [22].

Having introduced the experimental cell layout and the connected devices, the next section describes in detail the multi-modal software architecture and the involved modules which are required to perform the actual joint assembly task.

3. Software architecture and connected modules

In this section the software architecture and connected modules are introduced. Due to the requirements of an on-line robot motion control in the hybrid assembly cell, a real-time capable software architecture was implemented. The different functionalities of the system were integrated in a modular way, resulting in a reduced complexity of the involved agents. In Fig. 3 an overview of the developed system architecture is given. The communication backbone – connecting agents, sensors and actuators – is constituted by applying two middlewares: The Internet Communication Engine (ICE) [23], which is used for distributed computing and asynchronous message passing under topics of interest (e.g. speech and force–torque-data). The second communication backbone is the Real-Time DataBase (RTDB) [24], which is used as a local sensory information buffer. This ring-buffered database is using shared memory, to allow multiple modules accessing the same information without blocking effects.

3.1. Multi-modal in- and output

In order to make the communication with the system more intuitive and ergonomic for the worker, multi-modal interaction channels were established. The input channels feature speech, gaze and tactile input, whereas the system feedback is presented using a projected user interface and synthesized speech. These modalities are delineated in the following.

Speech input. To interact naturally and intuitively with the system, speech commands can be applied. With a speaker independent speech recognition, based on phonemes, the systems are able to understand English or German language. This makes it easy for other – probably untrained – workers to use the system. The speech input is captured non-invasive using a desktop microphone. To improve the recognition rate, the speech grammar is reduced in accordance to the current task context. Therefore, each connected module features its own grammar encoded in *Backus-Naur-Form* [25], which are provided to the knowledge-based system controller (see Section 4). Depending on the current processing rule – i.e. the current task – the required grammar is changed at runtime.

Gaze input. The workers gaze information is extracted using head-worn tracking glasses. These glasses enable a higher precision in the gaze tracking compared to remote eye tracking at the cost of being less comfortable and more invasive. Two cameras (one for each eye) are used to detect the bearers pupil. A third camera is used to record the scene for acquiring the transformation

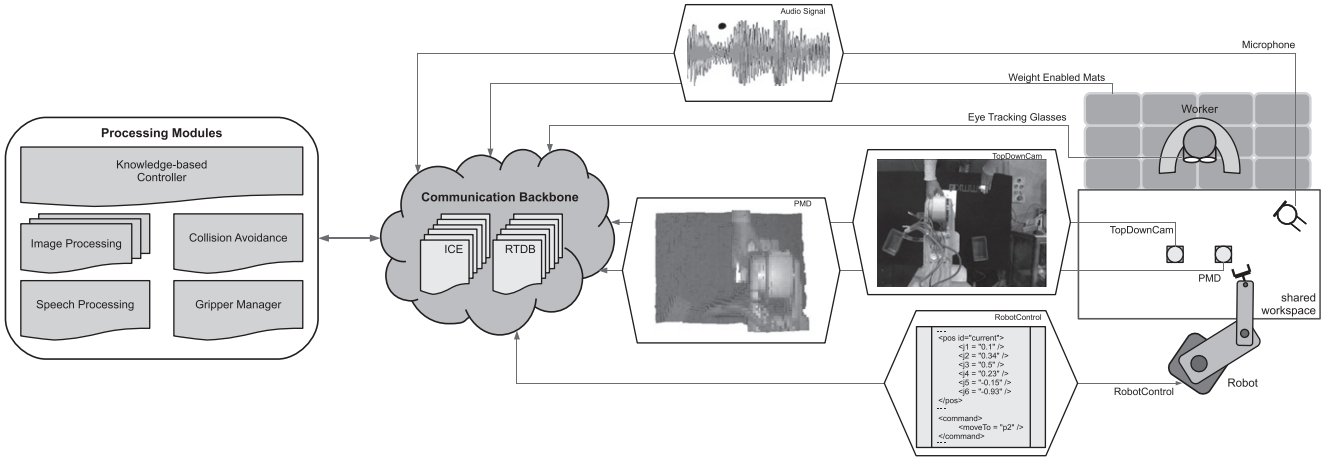

Fig. 3. Schematic overview of applied multi-modal system architecture (left to right): processing module, communication backbone, sensors and actuators, shared workspace.

matrix between the head related gaze vector and the world coordinate system to extract gaze in space information. More details on the used hardware and methods for the gaze-based interaction can be found in [26]. This gaze information provides a cue for the current focus of attention of the user. This focus of attention can be used either to directly activate projected sensitive interaction fields or select these and activate them by other modalities.

Tactile input. The last input modality is tactile interaction with the system, which is realized using real hardware buttons and displayed sensitive fields. These sensitive fields, the so-called *Soft-Buttons*, are projected onto the workbench and can be adapted in position and content during runtime. Thereby, they provide a very flexible and ergonomic interaction method, which allows to visualize information in the workers field of view. The location of the worker's hand is extracted using skin-color based image processing algorithms for segmentation (see [27] for detailed information). Together with this information and the position of the Soft-Buttons, a further module generates events, when a hand remains hovering over these sensitive fields for a certain amount of time.

Output and feedback channels of the system. To provide the worker with feedback in the audio domain, the system generates information using speech synthesis (TTS). This can be used to inform the worker about next steps in the assembly process or occurred errors, e.g. when the gripper manager (cf. Section 3.5) cannot find the required tool in one of the tool ports. The auditory channel can also be obeyed to inform the worker about the next action the robot is going to perform.

Additionally, the visual channel is obeyed to project assembly instructions and Soft-Buttons directly onto the workbench. This display technique allows also to project two dimensional projections of Cartesian trajectories (e.g. glue-lines) onto the workpiece being processed.

Supplemental research on the set-up has been conducted to make the robot motions more human-like. Thereby allowing the worker to predict the next movements of the robot, resulting in a smoother interaction process. Results on these human preferences can be found in [28].

3.2. Box tracking

During a hybrid assembly task, a typical action is fetching parts required for the next work step (see Section 5). In this case, these parts are kept in small storage boxes, which can be delivered to the work station via a conveyor belt (see Fig. 1). Therefore, the detection and tracking of those boxes has an important impact on the assembly task.

During a construction task, it is likely, that several boxes are located on the workbench. To fetch the correct parts at the right time, the system has to know, what parts are in which storage boxes. Therefore, in an initial phase, it is required to detect these boxes first, which is done by analyzing the acquired image from the top-down view camera (see Fig. 2).

In the here described set-up, different boxes are used, varying in size and color. To detect their position in pixel coordinates, a color-based image segmentation in the HSV-color-space is performed on the captured image [29]. To determine whether these areas are boxes, the extracted regions, called *blobs*, have to match predefined geometrical features. The center of gravity of the encircled pixel blob is calculated as a further geometrical feature for each box, see Fig. 4.

During runtime of the system, it is necessary to track the boxes in the image plane. The afore introduced procedure is a pure detection task, which cannot resolve the mapping of box to content. To get a consistent match between box and content, a further tracking component needs to be integrated and is introduced in the following.

Fig. 4. Result of the box tracking module: two red boxes are tracked. Each box is enclosed with a colored rectangle. The center of gravity is calculated for each box and drawn as a dot in the corresponding box color. Additionally, the tracking ids are indicated with colored circles. The content of each tracked box is depicted as a label. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In this set-up, optical flow analysis is applied to perform the tracking of the boxes. As feature points, the center of mass of each storage box is used. As implementation, the well-known algorithm of the Lucas Kanade Feature Tracker is applied, using the pyramidal implementation method. Due to the fact, that the content of the box is not of uniform color (cluttered content – screws, cables, etc.), it is sufficient to use only *one* feature point per box. This makes the implementation of the observing unit very fast and real-time capable.

The content of the box is stored in a mapping database, containing the current detected *boxID* and the corresponding *trackID* of the tracked feature point. To extract the actual content information, it is only necessary to match the *trackID* with the *boxID* of interest. If the user or the system requests a specific part stored in a box, the mapping of both ids is updated to retrieve the actual box position in world coordinates from a dedicated content.

The content can be initialized in two ways: at system startup, boxes already on the workbench are initialized with their contents by simply performing a pointing gesture towards the box.

During runtime, after fetching a storage box from the conveyor belt, the system knows the world position of this box and its content. Now, the database, which allocates content to *trackID*, only has to be updated with this information.

The automatic recognition of the box coordinates and its contents allow the user to freely position the boxes on the workbench. This is an important factor for the ergonomic layout of the workbench – the worker can decide where he wants to place the boxes. Furthermore, the system is able to follow the rearrangement of storage parts and can then automatically clear the table from obsolete storage boxes.

3.3. Collision avoidance controller

In a shared workspace, collisions need to be avoided for static (i.e. storage boxes and the workbench) and dynamic (i.e. the human and moving obstacles) objects in the shared space. In the developed collision avoidance controller, the avoidance is done in a reactive way using a dynamic internal 3D environment model as shown in Fig. 5. The internal 3D representation of the surrounding of the robot listens to all modifications including adding, updating, and disappearing of objects that were detected from input sensors (Fig. 6). This information is provided via a defined communication channel (*scene modification* that distributes information about changes in the environment. Therefore, new sensor

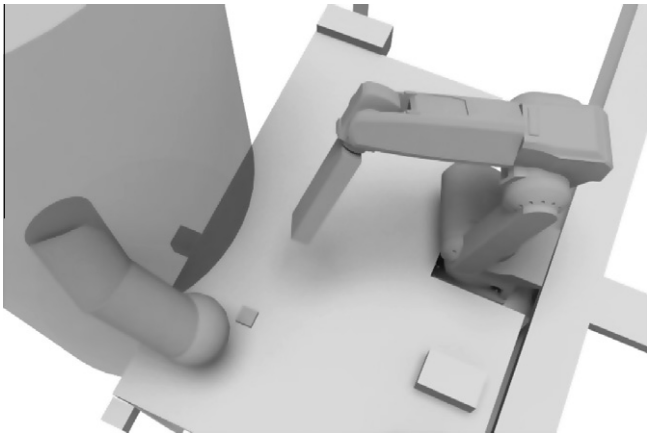

Fig. 5. Internal 3D environment model used in the collision avoidance module of the robot controller to compute distances and virtual forces that repel the robot. The cylindric shape approximates the human co-worker.

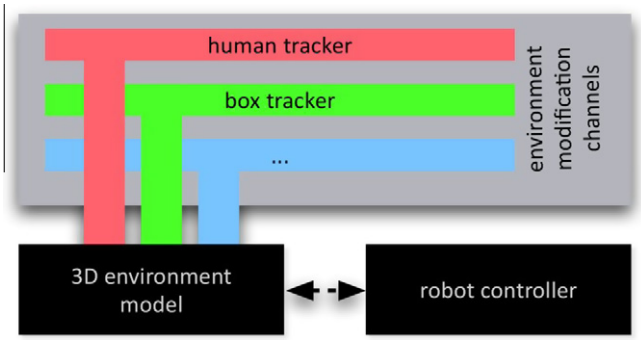

Fig. 6. The internal 3D environment model used in the collision avoidance module of the robot controller listens to all communication channels that provide information about modifications in the workspace of the robot.

modules can easily be added without changing anything in the robot controller.

The main challenge that arises here, is that the planned motion[2] and the avoidance motion must be handled in a way where they do not interfere with each other. Therefore, potential field methodology is fused to repel the robot from the obstacle with a constraint least-square optimization, that restricts the motion of the robot to safe orthogonal subspaces of the collision avoidance.

3.3.1. Virtual forces

To compute the velocity that repels the robot from surrounding obstacles, the minimum distances of all objects in the environment model (including self-collision) to all body parts of the robot need to be calculated. Fig. 7 depicts the body parts of the used robot in different colors along with an example of the minimum distances d_i (red lines) from an obstacle to a given joint configuration.

Opposite to simplified and only approximated models of manipulators (e.g., used in the skeleton algorithm presented in [30]), the distances of arbitrary shapes to a convex version of the real CAD-model of the robot are measured in order to reach a high precision of the virtual forces. With an efficient implementation of the Gilbert–Johnson–Keerthi algorithm [31], these distances are computed faster than the update rate of the robot controller.

After calculating the minimum distance vectors $v_{x,i}$ in Cartesian space (i.e. the direction of the applied virtual force), these need to

[2] I.e. moving from point A to point B in Cartesian space.

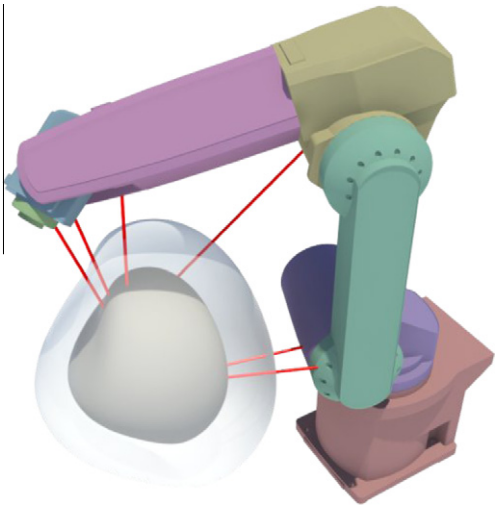

Fig. 7. Computing the repelling forces of an obstacle: The red lines illustrate the minimum distances of an obstacle to the body parts of the robot in a given joint configuration. If a distance is below a chosen security threshold (transparent bubble), the distance is used to compute virtual forces on the robot using potential fields. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

be transformed to velocities in joint space and the overall motion of the robot to avoid the collision has to be found. This is done according to

$$\underline{\dot{q}} = \sum_{i}^{I} \underline{\dot{q}}_i = \sum_{i}^{I} J_{P_r(i)}^T \cdot U_{rep,i}(q) \cdot v_{x,i}(q), \tag{1}$$

with I being the number of bodies of the robot, the current joint configuration of the robot q, the Jacobian of the minimum distance point on the robot $J_{P_r(i)}$ and the repelling potential function $U_{rep,i}$

$$U_{rep,i}(q) = \begin{cases} \frac{1}{2}\eta_i\left(\frac{1}{d_i(q)} - \frac{1}{Q^*}\right)^2 & \text{if } d_i(q) \leqslant Q^*, \\ 0 & \text{if } d_i(q) > Q^*, \end{cases}$$

with Q^* being the distance at which the potential field function is applied (see transparent bubble in Fig. 7).

3.3.2. Constraint least-square minimization

To ensure that the lower priority task is *projected* into an orthogonal subspace of the collision avoidance task, the mathematical framework of quadratic programming [32] is applied to minimize the quadratic error between optimal velocity of the lower priority task subject and the constraints of the higher priority task. The low-priority task execution ($\underline{\dot{q}}_{in}$) is optimized regarding *must have* constraints of the higher priority task. The projection is described according to

$$\min_{\underline{\dot{q}}_{in}} ||J_e \cdot \underline{\dot{q}}_{in} - \underline{\dot{x}}_t||^2, \tag{2}$$

where $\underline{\dot{x}}_t$ is the ideal linear and angular velocity to solve the lower priority task subject to the linear constraints of the form

$$C^T \underline{\dot{q}}_t \geqslant 0 \tag{3}$$

with the constraint matrix $\underline{C}^T$

$$\underline{C}^T = \begin{pmatrix} \underline{\dot{q}}_1^T \\ \vdots \\ \underline{\dot{q}}_I^T \end{pmatrix} \tag{4}$$

and $\underline{\dot{q}}_i$ calculated according to (1). This means only those velocities are valid, that are orthogonal to the direction of the collision

avoidance velocities or point in a direction that leaves the defined safety region. The output of the minimization process is then equal to the constrained joint velocity $\dot{q}^*$. Quadratic Programming in combination with collision avoidance on the level of joint acceleration was used in [33,34]. For a more detailed description of the applied algorithm, please refer to [35]. This shows a method, of how collision avoidance can be realized during motion planning within the robot controller.

3.4. Shared workspace surveillance

When human and robot collaborate in a shared workspace, safety critical situations can occur and these situations can be very harmful to the human, especially when standard industrial robots are used [36–40]. To increase the safety in the set-up, a redundant component for the shared workspace surveillance is introduced in this section.

The industrial standard EN ISO 10218-1:2006 [41] limits the maximum speed of an industrial robot in the collaborating mode to 250 mm/s, in case the robot is not sufficiently limited in power and force by inherent design. In this application, the robot should actively co-operate with the human worker. In accordance with existing regulations, this is not yet possible and therefore a new way of safe interaction needs to be discovered.

In Section 3.3, a robot control strategy was introduced that is capable of avoiding collision while trying to fulfill the task as good as the collision avoidance strategy allows. In cases of concrete danger for the human – i.e. direct contact between man and machine – the following additional safety module can stop the robot at any time. This module not only follows the communication channel based design principle, but also has a direct connection to the robots emergency shutdown.

For the detection of the human, different kinds of sensors can be used. A matrix of weight enabled mats (safety shutdown mats) on the floor can detect a contact with the worker's feet. Within this low resolution matrix, a rough estimation of the human worker's position can be extracted to infer whether a person is in front of the industrial robot or not [42]. At this point, the detection and localization of the human worker's arms and hands in the shared workspace has to be performed with an additional surveillance component. Therefore, a Photonic Mixer Device (PMD) is monitoring the shared workspace, which is divided into three regions: one area in which the industrial robot is engaging, another area engaged by the human worker. The overlapping region is the collaboration area.

In Fig. 8 a handover scenario in the real production cell and a visualization of the perceived sensory data is depicted. The industrial robot (number 1 in Fig. 8) is passing a box to the human worker (number 2), the human worker is detected both in the *worker workspace* and in the *collaboration area*. The rough position of the worker is seized (see Fig. 8 number 4) with the mats and marked as yellow rectangles (lower right image). The observation of the working table with the PMD camera is shown in Fig. 8 number 3 right. The visualization of the *dynamic passages* is not detecting any contact between the robot arm (number 1) and the workers hand (number 2). The black field in the visualization window indicates correctly that the worker and the robot are in the same static raster field. Thus, a case of contact is present, denying the robot further movements, until the contact is cleared.

To enhance the surveillance area, it is reasonable to integrate additional PMDs. These can be used to avoid occlusions within the shared workspace and to monitor regions not covered by the weight enabled mats.

3.5. Gripper manager

The final component described in this section is the gripper manager, taking care of the several tools (diverse grippers, pen, camera, and screwdriver) available in the set-up. Each of these tools can be uniquely identified by the robot via a coding pattern connected read out by the robot controller. Thereby, the robot knows always what tool is in its hand and adapts the kinematic parameters (tool-length and tool center point) according to the active tool.

The tools are placed in a port with six slots giving feedback about occupancy. After starting the system, the mapping of the manipulators to port slots is either unknown or might have changed since the last activation. Therefore, if a tool is needed during a task, the robot searches the tool in the occupied slots and saves the mapping. This mapping gets updated with every new recognized tool or when a slot is occupied with a tool different from the saved mapping.

The robot is connected with its skills to the knowledge-based system controller, as depicted in Fig. 3. If a certain skill gets requested, the gripper manager takes first of all care that the correct tool is in the robots hand, e.g. if the next action is *drill-mode*, the gripper manager checks the current tool in the hand and initiates, if necessary, a changing sequence towards the required tool *screwdriver*.

4. Knowledge-based system controller

In the previous section, the general system architecture and its connected modules were introduced. Here, the knowledge-based system controller for handling the complex processes within a hybrid assembly task is delineated.

An expert system was chosen, which uses first order logic processing rules and facts to handle the system work flow. Therefore,

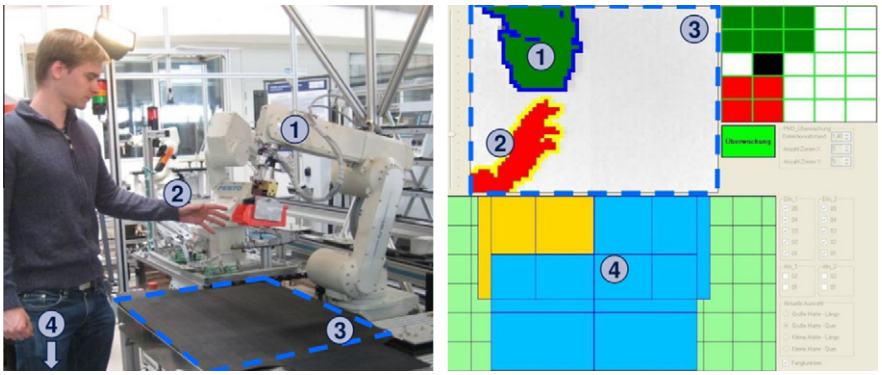

Fig. 8. Safety system in real production cell (left) and segmentation of human and robot based on depth image acquired by PMD camera above workbench (right).

JESS [43] is applied, which is implemented in Java and thereby allows a fast extension of its provided functionalities. As it is an interpreter, the functionality (new rules or modules) can be extended during runtime. JESS is operating on a working memory containing the facts and rules of the system. Every change (creation, deletion or update) of a fact within the working memory results in checking, if there is a rule matching the changed conditions. Furthermore, it fits perfectly into the developed communication backbone, which is applied to update events and communicate with the connected modules (see Section 3). In the following, a closer look into relevant facts and rules within the presented system is given.

4.1. Facts

The facts of the system provide the controller with necessary information about occurring events, the environment and available skills of the connected modules. These facts – coming from the communication backbone – are transformed into predicates to be processed by the rule engine, which will be explained in Section 4.2.

4.1.1. Events

One subset of the facts are events. These can be for example status changes of soft-buttons, results of the speech recognition unit, etc. If the worker activates for example soft-button two, the working memory of the system controller is updated with the fact:

```
(soft-button
  (id 2)
  (status "activated")
)
```

Now all existing rules listening to this event are triggered. After the event has been processed it is retracted from the working memory.

4.1.2. Environment information

Information about the environment is also represented as facts. This covers box data, worker's hand positions, etc. For example, a red box is represented with the following notation:

```
(object
  (type "box")
  (description "screws")
  (color "red")
  (position x y z alpha beta gamma)
)
```

In this case the position data contains the Cartesian coordinates within the world coordinate system as well as the rotational information about the objects orientation. Environmental information (e.g. worker's hand position) must have a very high update rate to provide the collision avoidance and the shared workspace surveillance components with the required data.

4.1.3. Skills

Every connected module can report its skills in form of facts. The robot connects to the knowledge-based system controller and registers only its available skills. These skills include several basic blocks with actions, e.g. *move to position, open gripper*, and several higher-level skills including picking up an object from the table. The system controller has no clue about the real hardware, but it knows what the hardware can do and can access these abilities. As depicted in Fig. 9 these abilities are much more complex than the afore introduced basic blocks. These higher-level skills are transformed into atomic operations and then executed on the real hardware, e.g. for the command *give* would result in the following operations:

- move to the hand over position
- wait for force
- open the gripper
- move away

Furthermore, pre- and post-conditions are given to define transition between states:

```
(skill
  (agent ?robot)
  (command "opengripper")
  (precond "gripper closed")
  (postcond "gripper open")
)
```

These can be used to automatically fill missing intermediate steps between the atomic operations.

4.1.4. Tasks

The introduced (high-level) skills can be combined to reach a goal. One possible sequence of such a combination is called a task. Each task has a unique name which is an identifier for the task itself and is used by the worker to call it for execution. The order in which the skills are processed, is defined by an increasing index. A sample task consisting of two steps, namely move the robot in a parking position and turn off the signaling lights would be:

```
(task
  (id 0)
  (name "task 1")
  (command "(?robot moveaway)")
)
(task
  (id 1)
  (name "task 1")
  (command "(?signalmanager lightsOff)")
)
```

A special form of a task is named *learning* and enables the system to extend the set of available tasks. The learning of such a new task consists of teaching the correct sequence of actions. These actions can be either atomic operations or previously learned tasks to build up more complex sequences.

In the first step, a temporal task is constructed, which serves as placeholder for the task to be taught. Rules construct the speech recognition grammar from the available set of skills and thereby prepare the system for the instruction based learning. Now the worker can add new steps to this task in a loop by giving voice based instructions ("fetch red box", etc.). The learning task is finished when the user saves the task by assigning it an unique name. A successful execution of the learning task, results in adding a new element to set of available tasks. The learned tasks are stored in a XML-representation into a file to generate a persistent task database over time and enhance the systems capabilities.

The transition between the sequence of actions is performed by the rules for task execution, which are explained in more detail in the next section.

4.2. Rules

The second important component of the knowledge-based system controller is the capability of processing complex rules

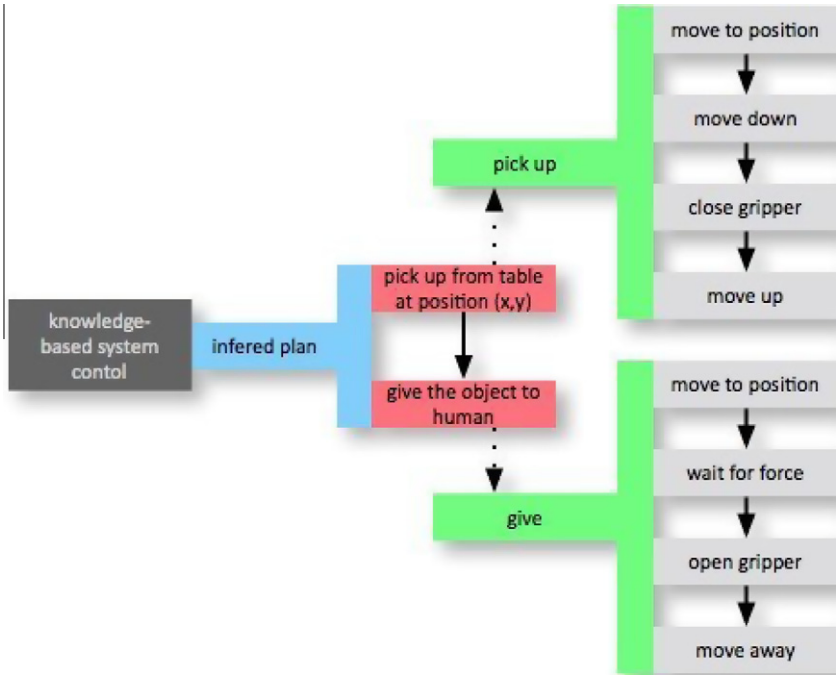

Fig. 9. How inferred plans lead to a robot motion.

describing the system work flow. As introduced, changes in the working memory can activate corresponding rules matching current conditions.

4.2.1. Event handling

One subset of the rules is applied to handle events. These rules process the events, which are provided by the perceiving modules, i.e. *button pressed*, *box found* or *speech input*. For example, if the worker is pressing a displayed Soft-Button, the event-handler evaluates the corresponding *id* and modifies the status flag of the linked status fact. This changed status results in the activation of a linked processing rule.

4.2.2. Task execution

The processing rules for the task execution are "innate" knowledge, which tell the system how to instantiate a new task. Therefore, the task that shall be executed is copied into a running task instance. The execution of this running task starts at *index 0* and executes the associated command on the given agent. These command are executed in a non-blocking operation mode. Therefore, the execution status is managed in an attribute, which can have one of these values (see Fig. 10): *todo*, *starting*, *processing*, *aborted*, *failed*, *completed*.

The initial status value is *todo*. The status value is changed form *todo* to *starting*, when the system controller has decided to activate the command on the agent. Now, necessary initialization routines can be prepared before the real processing starts. After this initialization was successful, the status is switched to *processing*. In case of an error during initialization, the status is switched to *failed*.

The system controller then waits for the execution result of the command. In case of an error during the processing, the status would be changed to *failed*. Furthermore, it is possible to interrupt a running command, resulting in the *aborted* status. If the status is *aborted* or *failed* an appropriate exception handling is required. Else, the successful completion of the command is indicated with the status attribute set to *completed* by the agent.

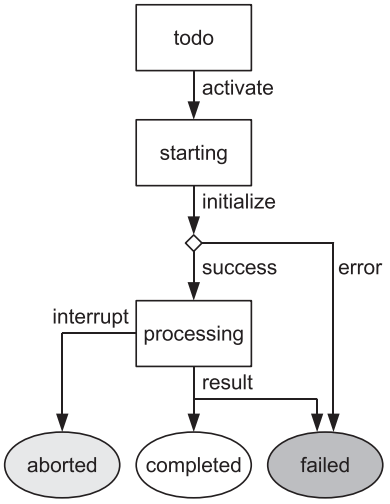

Fig. 10. Schematic overview about the execution status of the agent.

5. Application scenario

In this section an application scenario is introduced, in which the worker wants to assemble a product, which is unknown to the system. Thus, the worker has to teach in the required assembly steps.

A sample dialog for this scenario between the worker `W` and the robot `R` is presented below:

```
(0) W: I want to teach you a new task.
(1) R: Okay. Ready when you are.
(2) W: Fetch red box.
(3) R: Added new step "fetch red box".
(4) W: Give object.
(5) R: Added new step "give".
(6) W: Go to drill-mode.
(7) R: Added new step "drill-mode".
```

(8) W: Save this task as "task four".
(9) R: Task saved as "task four".

In (0), he is launching the learning task (cf. Subsection 4.1.4) with the phrase "I want to teach you a new task." After the learning task has been successfully initiated and the speech recognition grammar has been adapted, the system is informing the worker with (1). From now on, the worker can add next steps via voice based instructions ((2), (4), (6)), which are appended to the task. The system is informing the user via text-to-speech (TTS) about the successfully appended commands ((3), (5), (7)). In case a wrong production step has been added, the user can say "Undo last step" to remove the last uttered command from the list. The learning task is completed by the user assigning a name to the task ((8)). This task is stored, which is confirmed by the system (9). From here on it is also available for other workers.

Task execution is triggered with the speech command "Let us start task four". Now, the knowledge-based system controller tries to locate the specified task in the fact set. In the given case, this search will be successful and the task is instantiated (cf. Section 4.2.2). The first command given in this example ("fetch red box") is executed. The knowledge-based system controller now refreshes the working memory for objects – which includes information from the box tracking module (see Section 3.2) – and tries to infer the position of an object from type box with a red color. Outcome of this unification process can be threefold: It can find no solution, because there is no red box; it can find one solution, one red box on the workbench; it can find multiple solutions, several red boxes.

If no or multiple solutions are found, further user interaction is required. In this case, the *pickup* skill (cf. Fig. 9) is activated, using the unified position data of the box tracking.

At this point, the robot agent takes control of the gripping process, signaled by changing the status of the *pickup* command to *starting* (cf. Section 4.2.2). During the initialization, the gripper manager (cf. Section 3.5) checks the currently mounted gripper, if it is the correct tool for the *pickup* command. If not, the manager tries to locate and fetch the correct gripper first. After having the right manipulator, the status is changed to *processing*. Once the robot has fetched the red box (status changed to *completed*), the next command *give* is executed resulting in the handover of the box. After the robot has reached the handover position, the information of the force–torque channel is subscribed to be able to detect when the worker touches the box. The published data are monitored and finally resulting in the opening of the gripper. Now the box is free and the *give* command finishes with the robot moving out of the shared workspace.

The sample task "task four" finishes with the "drill mode" operation in a similar way to the previously described steps. This concludes the presented sample application. The completed task is retracted from the working memory and thus, the hybrid assembly cell is ready for the next task.

6. Discussion

6.1. Cell layout

The described cell layout was selected to force interactions between the human worker and the industrial robot. Therefore, regions were created, which are only accessible for one of the interactions partners (e.g. only the robot can pick up objects from the belt). A further characteristic of the small shared workspace is the higher appearance of possible collisions, because overlapping trajectories of human and robot are very likely to occur. Furthermore, the size of the workspace is small enough to be monitored with the mounted cameras.

However, the size of the workspace limits on the other hand the amount of possible operations, which can be performed jointly by human and robot. Huge parts cannot be assembled, because of the restricted movement range of the robot.

Due to the absent mobility of the robot and the required sensory safety equipment, the cell has to be set up at a fixed location. Thereby, it is most beneficial for installing it in factory environments.

Finally, the tool change station allows to perform various manipulation- and handling tasks at the cost of changeover times. Here, an anthropomorphic hand would allow more flexibility, at the cost of a more complex grasping manager replacing the gripper manager.

6.2. Software architecture

The presented system covers aspects from low-level data processing up to higher-level reasoning. For each of these domains different programming paradigms and languages have evolved. Therefore, a middleware is applied, which allows to port interfaces –written in a meta-language– into the respective languages. Furthermore, the set-up features heterogeneous sensors and actuators.

Each of these components and software-modules are represented as agents. The software architecture was constructed accordingly to support an agent-based concept, where the modules are connected with each other and report their skills to a central system controller orchestrating the services. These skills represent higher-level interfaces towards the implemented functionalities of each module. With these skills, the controller is capable of solving a task by combining them to create more complex operations.

As the robotic controller requires position updates every 7 ms, it is essential to process the sensory data and calculate the new trajectory within this small time span. This real-time constraint induces the necessity to compute and transmit the perceived input on-line. Therefore, the processing power of multiple PCs has to be combined. Thus, a middleware was selected to support this distributed architecture.

Due to the large amount of modules and computing nodes involved, it is reasonable to keep the traffic on the communication backbone as low as possible. Therefore, the modules process the low-level sensor data and generate higher level information, which is then published via the communication channel. Finally, the applied middleware supports redundancy and scalability. Thus, redundant modules can run on backup nodes, which take over in case of failures. New modules can be easily integrated into the communication backbone. These have to implement the unified interfaces, which allow communication with the rest of the modules.

6.3. Multi-modal input

One goal of the described system is to support the multimodal interaction with the user. Therefore, speech, gaze and haptic is provided as user input channels.

For the speech channel, a command word based speech recognizer was integrated. This allows a more robust and user-independent recognition of utterances. The major drawback of this approach, is that it requires a reduced grammar for the allowed sentences. Of course, natural language understanding would greatly increase the intuitiveness of this communication method. On the other hand, taking into account, that the dialog between robot and worker in a factory will be based more on instructions and less on natural conversations, the selected approach is suitable for the presented application. As the speech recognition grammar is

reduced to the current context, it is thereby more robust in the noisy factory environment.

For operating the projected soft-buttons hands-free, gaze information has been provided as further interaction method. As introduced, the worker has to shift his visual attention onto the button to be selected. The applied tracking glasses work with very high precision, but are not comfortable to be worn over a longer period of time. A remote gaze-tracking system would compensate this lack in ergonomics and shall be integrated in the future.

The final modality is haptic interaction via projected soft-buttons. This interaction method allows to make an industrial surface sensitive for user inputs by using a top-down projection unit and a camera. In consequence of this set-up, the projection may have occlusions, which are caused by the robot moving through the display beam. A different approach would be to project the information from beneath the workbench. Thereby, occlusions from above could be avoided, but this method would require a non-standard surface (e.g. translucent glass) on the workbench. Additionally, this method would not allow the projection of trajectories onto the workpiece.

6.4. Collision avoidance and shared workspace surveillance

To actually bring human and robot together in the same workspace is a challenging task. Especially the area of the workspace that can be accessed by both the human and the robot needs to be monitored with reliable and robust sensors. Depth sensors based on the time-of-flight principle are a good solution for this, because they have a high resolution in time opposite to the disadvantage of low spacial resolution. But in the case of measuring the distances of human and robot, this is sufficient. As presented in Section 3.3 repelling forces computed via artificial potential fields are used to avoid collisions of robot and human. One drawback of the applied potential field method is the fact, that the robot can stuck in local minima. Another issue is, that the usage of only the minimum distance between robot body and obstacles can lead to an oscillatory behavior between obstacles. This can be solved, if multiple distances and therefore multiple forces for each robot body are used. Along with the hierarchical structure of the controller, higher level behaviors of the robot become possible.

6.5. Knowledge-based system controller

As introduced previously, modules offer their functionality as skills. The orchestration of these skills to execute tasks is done by the system controller. Therefore, this controller uses the previously instructed knowledge about how to solve a task. This task is represented on different levels of abstraction, where each subsequence can be split up into more granular operations up to actual system operations. With this predicate-based representation it is possible to infer and reason about the combination of skills to reach the desired goal. Thereby, this type of controller allows more flexibility than a finite state machine, because new transitions can be added via spoken instructions or inferred autonomously. These advantages justify the higher implementation efforts for designing the common skill-based representation.

However, the more available skills have to be considered for solving the task, the more time or processing power is required. This obviously leads to increasing reaction time on new events. To overcome this drawback, the components which are relevant for safety have an additional direct communication channel to the robotic controller. Thereby, they can bypass these potential processing delays to ensure fast response times.

Within the presented application scenario, presented in Section 5, this method of controlling the system's execution flow has shown promising potential for the hybrid assembly, which needs to be evaluated in the ongoing research work.

7. Conclusion

In this article an overview about an approach towards a teachable hybrid assembly system was presented. The introduced system uses a knowledge-based system controller that is orchestrating the skills of connected modules and representing sensor events and the work tasks as first order logic predicates, thereby allowing a modification of the knowledge (rules and facts) during runtime. An experimental hybrid assembly cell inside a Cognitive Factory was presented mostly basing on regular and standardized industrial hardware components. The software to control these hardware components (sensors and actuators) was developed in a modular way and integrated into a complex software architecture, which allows real-time processing. Furthermore, it is capable to process voice, gaze and tactile interaction channels. In addition, the combination of the shared workspace surveillance unit and the collision avoidance module provide the possibility to share the workspace between the human and the robot at the same time. Finally, the interaction process between the different presented modules were outlined using an example application scenario.

The presented components show opportunities towards increasing safety and interaction in a hybrid assembly cell, in which a human and an industrial robot work together on a joint task. Additionally, even workers without robot programming experience become able to teach-in novel tasks to the system using instructions.

Future work will concentrate on an improved ergonomic view of the assembly process and evaluation of the presented safety concepts. Therefore, the observation and interpretation of the worker's activities will become one focus of the ongoing research as suggested in [44]. Furthermore, modules for a precise 3D recognition of the workpiece and its components will be integrated. This will allow for a semantic understanding of the production process from the point of view of the hybrid system and will enable an additional optimization of the assembly sequence. After repeated observations of what was added at what position and at which time, the system will be able to collect knowledge on a semantic level of the process. This combination of an improved worker surveillance together with a better understanding of the scene may make today's production facilities in high loan countries more competitive again.

Acknowledgments

This ongoing work is partially supported by the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see www.cotesys.org for further details.

References

[1] C. Parlitz, C. Meyer, PowerMate – Schrankenlose Mensch-Roboter-Kooperation, <http://www.ipa.fraunhofer.de/Arbeitsgebiete/robotersysteme>, 2005.
[2] E. Helms, R.D. Schraft, M.Hägele, rob@work: Robot assistant in industrial environments, in: Proceedings of the 11th IEEE International Workshop on Robot and Human interactive Communication, Berlin, Germany, 2002, pp. 399–404.
[3] G. Reinhart, W. Vogl, W.Rösel, F. Wallhoff, C. Lenz, JAHIR – Joint Action for Humans and Industrial Robots, Intelligente Sensorik – Robotik und Automation, June 2007.
[4] G. Reinhart, W. Tekouo, W. Rösel, M. Wiesbeck, W. Vogl, Robotergestützte kognitive Montagesysteme – Montagesysteme der Zukunft, wt Werkstattstechnik online 97 (9) (2007) 689–693.
[5] A. Kochan, Robots and operators work hand in hand, Industrial Robot: An International Journal 33 (6) (2006) 422–424.

[6] World robotics – executive summary 2009. URL <http://www.worldrobotics.org/downloads/2009_executive_summary.pdf>.
[7] R.D. Schraft, C. Meyer, C. Parlitz, E. Helms, PowerMate – a safe and intuitive robot assistant for handling and assembly tasks, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 4074–4079.
[8] Mrk-systeme gmbh, URL <http://www.mrk-systeme.de>.
[9] Project site of smerobot, URL <http://www.smerobot.org>.
[10] J. Colgate, W. Wannasuphoprasit, M. Peshkin, Cobots: robots for collaboration with human operator, in: Proceedings of the ASME Dynamic Systems and Control Division, vol. 58, 1996, pp. 433–440.
[11] M. Akella, M. Peshkin, E. Colgate, W. Wannasuphoprasit, Cobots – a novel material handling technology, in: Proceedings of the ASME Winter Annual Meeting, 1998.
[12] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöllner, M. Bordegoni, Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm, 1999.
[13] O. Rogalla, M. Ehrenmann, R. Zöllner, R. Becher, R. Dillmann, Using gesture and speech control for commanding a robot assistant, in: IEEE International Workshop on Robot and Human Interactive Communication, Piscataway, IEEE Press, 2002, pp. 454–459.
[14] R. Schraft, C. Meyer, The need for an Intuitive Teaching Method for Small and Medium Enterprises, VDI-Bericht 1956, May 2006.
[15] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, K. Ikeuchi, Leg task models for reproducing human dance motions on biped humanoid robots, Journal of the Robotics Society of Japan 24 (2006) 388–399.
[16] M. Skubic, R. Volz, Acquiring robust force-based assembly skills from human demonstration, IEEE Transactions on Robotics and Automation 16 (2000) 772–781.
[17] M. Zäh, C. Lau, M. Wiesbeck, M. Ostgathe, W. Vogl, Towards the cognitive factory, in: International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV), Toronto, Canada, 2007.
[18] M. Amoretti, M. Reggiani, Architectural paradigms for robotics applications, Advanced Engineering Informatics 24 (1) (2010) 4–13.
[19] W. Shen, Q. Hao, H.J. Yoon, D.H. Norrie, Applications of agent-based systems in intelligent manufacturing: an updated review, Advanced Engineering Informatics 20 (2006) 415–431.
[20] J. Ota, Multi-agent robot systems as distributed autonomous systems, Advanced Engineering Informatics 20 (2006) 59–70.
[21] K. Domdouzis, B. Kumar, C. Anumba, Radio-Frequency Identification (RFID) applications: a brief introduction, Advanced Engineering Informatics 21 (2007) 350–355.
[22] M.F. Zaeh, M. Ostgathe, C. Lau, M. Wiesbeck, Adaptive, product-based control of production processes, in: M.F. Zaeh (Ed.), rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2009), ElMaraghy, H.A., München, 2009, pp. 99–110.
[23] Zeroc, Internet Communications Engine, <http://www.zeroc.com>.
[24] M. Goebel, Real-time data base, <http://www.kogmo-rtdb.de>.
[25] D.E. Knuth, backus normal form vs. backus naur form, Commun. ACM 7 (12) (1964) 735–736. doi:http://doi.acm.org/10.1145/355588.365140.
[26] S. Bardins, T. Poitschke, S. Kohlbecher, Gaze-based interaction in various environments, in: Proceedings of First ACM International Workshop on Vision Networks for Behaviour Analysis, VNBA 2008, Vancouver, Canada, 2008.
[27] T. Rehrl, A. Bannat, J. Gast, G. Rigoll, F. Wallhoff, cfhmi: a novel contact-free human-machine interface, in: J. Jacko (Ed.), Proc. Int. Conf. on Human–Computer Interaction HCI 2009, San Diego, CA, USA, LNCS, vol. 5611, Springer, 2009, pp. 246–254, 19–24.07.2009, Human–Computer Interaction. Novel Interaction Methods and Techniques, ISBN: 978-3-642-02576-1.
[28] M. Huber, C. Lenz, M. Rickert, A. Knoll, T. Brandt, S. Glassauer, Human Preferences in industrial human-robot interactions, in: Proceedings of the International Workshop on Cognition for Technical Systems, 2008.
[29] A. Bannat, J. Gast, G. Rigoll, F. Wallhoff, Event analysis and interpretation of human activity for augmented reality-based assistant systems, in: Proc. IEEE Intern. Conference on Intelligent Computer Communication and Processing ICCP 2008, IEEE, Cluj-Napoca, Romania, 2008.
[30] A. De Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, G. Hirzinger, The skeleton algorithm for self-collision avoidance of a humanoid manipulator, in: Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2007, pp. 1–6, doi:10.1109/AIM.2007.4412606.
[31] G. van den Bergen, A fast and robust GJK implementation for collision detection of convex objects, Journal of Graphics Tools 4 (2) (1999) 7–25.
[32] R. Fletcher, Practical Methods of Optimization Part II: Constrained Optimization, second ed., John Wiley & Sons, New York, 1987 (Chapter 10: Quadratic Programming, pp. 229–258).
[33] E. Freund, M. Schluse, J. Rossmann, Dynamic collision avoidance for redundant multi-robot systems, in: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, 2001, pp. 1201–1206, doi:10.1109/IROS.2001.977146.
[34] E. Freund, J. Rossman, The basic ideas of a proven dynamic collision avoidance approach for multi-robot manipulator systems, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, 2003, pp. 1173–1177.
[35] C. Lenz, M. Rickert, G. Panin, A. Knoll, Constraint task-based control in industrial settings, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2009.
[36] S. Haddadin, A. Albu-Schäffer, G. Hirzinger, Safety evaluation of physical human–robot interaction via crash-testing, in: Robotics: Science and Systems Conference (RSS2007), 2007, pp. 217–224.
[37] S. Haddadin, A. Albu-Schäffer, G. Hirzinger, The role of the robot mass and velocity in physical human–robot interaction – Part I: unconstrained blunt impacts, in: Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, USA, 2008, pp. 1331–1338.
[38] S. Haddadin, A. Albu-Schäffer, G. Hirzinger, The role of the robot mass and velocity in physical human–robot interaction – PartII: constrained blunt impacts, in: Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, USA, 2008, pp. 1339–1345.
[39] S. Haddadin, A. Albu-Schäffer, M. Frommberger, J. Rossmann, G. Hirzinger, The "DLR Crash Report": Towards a Standard Crash-Testing Protocol for Robot Safety-Part I: Results, in: Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, pp. 272–279.
[40] S. Haddadin, A. Albu-Schäffer, M. Frommberger, J. Rossmann, G. Hirzinger, The "DLR Crash Report": Towards a Standard Crash-Testing Protocol for Robot Safety-Part II: Discussions, in: Proceedings of the IEEE International Conference Robotics and Automation, Kobe, Japan, 2009, pp. 280–287.
[41] EN ISO 10218-1:2006, Robots for Industrial Environments – Safety Requirements – Part 1: Robot, 2006.
[42] M. Zäh, W. Rösel, Safety aspects in a human-robot interaction scenario: a human worker is co-operating with an industrial robot, in: International Conference on Social Robotics (ICSR), FIRA, Incheon, Korea, 2009.
[43] Jess, the Rule Engine for the Java™ Platform, <http://www.jessrules.com/>.
[44] M. Hasanuzzamana, T. Zhanga, V. Ampornaramvetha, H. Gotodaa, Y. Shiraib, H. Ueno, Adaptive visual gesture recognition for human-robot interaction using a knowledge-based software platform, Advanced Engineering Informatics 55 (2007) 643–657.