Multimodal Sensor-Based Whole-Body Control for Human-Robot Collaboration in Industrial Settings

José de Gea Fernández[a], Dennis Mronga[a], Martin Günther[a], Tobias Knobloch[d], Malte Wirkus[a], Martin Schröer[b], Mathias Trampler[a], Stefan Stiene[a], Elsa Kirchner[a,b], Vinzenz Bargsten[b], Timo Bänziger[c], Johannes Teiwes[c], Thomas Krüger[c], Frank Kirchner[a,b]

[a] *Robotics Innovation Center at German Research Center for Artificial Intelligence (DFKI), Robert-Hooke Str. 1, 28359 Bremen, Germany*
[b] *Robotics Group of the University of Bremen, Robert-Hooke Str. 1, 28359 Bremen, Germany*
[c] *Volkswagen AG, Berliner Ring 2, 38436 Wolfsburg, Germany*
[d] *Hochschule Aschaffenburg, Würzburger Str. 45, 63743 Aschaffenburg, Germany*

Abstract

This paper describes the development of a dual-arm robotic system for industrial human-robot collaboration. The robot demonstrator described here possesses multiple sensor modalities for the monitoring of the shared human-robot workspace and is equipped with the ability for real-time collision-free dual-arm manipulation. A whole-body control framework is used as a key control element which generates a coherent output signal for the robot's joints given the multiple controller inputs, tasks' priorities, physical constraints, and current situation. Furthermore, sets of controller-constraints combinations of the whole-body controller constitute the basic building blocks that describe actions of a high-level action plan to be sequentially executed. In addition, the robotic system can be controlled in an intuitive manner via human gestures. These individual robotic capabilities are combined into an industrial demonstrator which is validated in a gearbox assembly station of a Volkswagen factory.

Keywords:

1. Motivation

Industrial manufacturing is undergoing major changes and transformations. New technologies are introduced by digitization; consumers are demanding that the manufactured products are produced in an increasing number of variants while the workforce's average age is shifted by the demographic change. All of these factors are important drivers for introducing workplaces where human and robot work together.

In recent years, a new generation of industrial robots is being deployed to allow the collaboration (and thus, the sharing of common spaces) between robots and humans. In general, those robots differ from the classical industrial robots in several aspects: (1) they do not need to work at the high speeds of the classical counterparts; the payloads do not need to be as high, but similar to the ones a human can carry, (2) they are not there to replace humans but to cooperate with them, (3) they need to be flexible to allow very short production series, thus they need to be simple to command and program and (4) they need to be safe to work sharing common spaces with humans.

There are additional technological aspects which are important for a company like the Volkswagen Group as it consists of multiple brands and operates multiple manufacturing plants worldwide. The new robotic system needs to be modular on a functional level since that allows reusability and scalability to different applications and use cases. This includes the vital perspective that the developed functional modules can be used for any robot coming from any manufacturer. Since the market for intelligent lightweight robots is developing rapidly, the modularization provides the opportunity to quickly integrate improved robot systems into existing plants. Also, certain functionalities like collision avoidance can be integrated into other applications.

In this context, there is still a largely-disregarded challenge, in robotics research in general, and in applications such as the one described here in particular, concerning the seamless integration of different functionalities in the same robot working in real-time and in a real-world scenario. There is a vast amount

of single functionalities shown in single laboratories and arguably most of the algorithms required to build complex intelligent robots are already out there, at least enough so that we should find more complex and intelligent robots in real applications. However, there is a clear mismatch between the complexity of the systems deployed in real scenarios and the availability of a vast collection of algorithms in all areas of robotics. In our opinion, one of the key 'research' questions is how to combine different controllers at different levels of the robot's architecture together in a coherent and holistic framework that augments the capabilities of the system beyond the sum of the single contributions. In this work, multiple functional subsystems, developed and validated in separate projects, are for the first time combined into an industrial demonstrator which addresses an actual automotive manufacturing scenario. Thus, the focus was not set on new safety or interaction concepts but on multiple sensor-based controllers and intuitive interaction methods and, above all, on enabling the integration of those different capabilities in a coherent framework in a real system and in a realistic scenario. Such a tight integration is only possible: (a) with a coherent software framework which is able to integrate, monitor, and supervise the different components at different levels of the control architecture (from the lowest level —driver side— to the highest level of interaction or cognition) and (b) by using core components such as whole-body control, which not only controls the different parts of the robot (i.e., redundancy resolution) and sensor data streams in a coherent manner but also ensures the stability of the robot control by finding an optimal tradeoff from the diverse and heterogeneous controller inputs.

More specifically, this work linked novel multisensor-based workspace monitoring and tracking algorithms with collision avoidance algorithms to create a robotic system with the capability of real-time avoidance of collisions both with itself and with external objects. A jacket was sensorized to be used as a gesture recognition device worn by the user in order to command the robotic system. Additionally, a set of whole-body controllers are used as building blocks that describe single actions of a high-level robot behaviour plan. And finally, a modular, robot-agnostic software control framework was used with two main

purposes: (1) for the seamless integration of all components in a coherent work flow, and (2) to allow reusing generic software components to describe a variety of complex manipulation behaviours, whilst keeping independence from the particular robot hardware.

In addition to the integration effort, there are a number of original research contributions in this paper, in the cases in which available technologies were not available or required modifications to be used in this context. Those are:

- the 3D tracking algorithms, which were completely developed in the framework of this work,
- the use of whole-body control, not only in its original intended use as a redundancy-resolution solver, but also as a coherent controller that ensures proper integration of very different controller types and sensor input modalities,
- a highly modular and reusable software framework together with an event-based task plan manager that provides a deterministic process control and the possibility to supervise and monitor all the running robot processes in real-time to create a reliable system, and
- the external real-time collision avoidance, which includes some particularities (robot-obstacle distances are evaluated directly in 3D space, which allows integration of multiple, disparate, sensing modalities that provide point cloud data, as it is shown here).

2. Related Works

The area of human-robot collaboration has experienced a significant increase of interest in the past years, first from the research community and, more recently, from the industrial community as well. The reason lies in key enabling technologies appearing in the market, probably most importantly a new generation of lightweight robots which incorporate different concepts (control software

or mechatronic design) to allow the interaction with humans while ensuring a certain degree of safety.

From the point of view of the robot itself, there are several examples of a new generation of "safe" robots which allow physical contact between humans and robots. All of them have been commercially available for a couple of years, and are generally affordable and flexible in its use.

- Rethink Robotics [8] offers the robots Baxter and Sawyer. The first one is conceived as a research platform and equipped with two arms and a tactile screen as head; Sawyer, on the contrary, is a single arm for industrial application. The key feature of both of them is their inherent safety to work together with humans: the motors incorporate in series an elastic element (mechanical spring) which ensures that even in case of software malfunctioning or power failure, the robot remains always flexible ("soft") to external contact.

- Universal Robots [10]. These robots look externally like traditional industrial robots but are certifiable for most human-robot collaborative tasks according to the current standards. In contrast to Baxter, the robots are not inherently safe, but include several measures described in the ISO 10218 standard: the possibility for speed reduction if external sensors detect a person in the proximity and, especially, the limitation of the maximum forces. The latter is achieved by a patented solution that uses motor currents to estimate the joint torques and thus the maximum forces at the end-effector without requiring any additional sensor.

- KUKA LBR iiwa [7]. These are the robots used in this project. They are an example of academic research brought to industrial product. For more than a decade, researchers at DLR developed lightweight robot arms [22] with extremely powerful actuators in relation to its relatively low weight and, for the first time, including joint torque sensors which enable the possibility of accurate dynamic control of the robot and, thus, of so-called

active compliance control. Those joint torque sensors, additionally, also provide immediate information about contacts between the robot and the environment, which can be used for collision detection and reaction. After a long development time between DLR and KUKA, the arms are nowadays sold by KUKA.

- Franka[13]. This robot was just announced in 2016 and will be available beginning 2017. As in the KUKA LBR iiwa, Franka also includes joint torque sensors and similar joint electronics to create a powerful and affordable lightweight collaborative robot.

Besides, nowadays almost every industrial robot manufacturer includes among its portfolio a "collaborative" version of its robots: Fanuc with its new CR-35A [12], ABB with the dual-arm robot Yumi [16], Yaskawa with Motoman HC10 [14] or COMAU with its high-payload collaborative robot AURA [11]. Most of these new generation of robots are designed to offer the maximum flexibility to be programmed in an intuitive way even by non-roboticists for a variety of tasks and thus the time to "teach" the robot to do a new task tries to be kept very short.

Even when the robot itself cannot be considered as a "safe" robot, the use of external sensors monitoring the shared human-robot workspace can enable the use of classical industrial robots in collaborative environments, so that there is no need to confine the robot behind fences anymore. From this environment perception point of view, there are different safety-rated technologies currently available.

- Safety laser scanners: The most frequently used safety sensors are safety laser scanners. Typically they are mounted in a way that the sensor forms a 2D-horizontal field of view around the robot. Most scanners have the possibility to define warning and safety fields. The scanner's safety fields are hardwired with an emergency switch of the robot. If an object enters one of the defined safety fields the robot is switched off. The most well-known manufactures of safety laser scanners are SICK AG, Leuze Elec-

tronic GmbH, HOKUYO AUTOMATIC CO., or OMRON. The safety integrity level (SIL) of these scanners is normally two or three.

- SafetyEYE [4]. The PILZ safetyEYE is a camera system for three-dimensional zone monitoring. The camera system is mounted up to 4 meters above ground giving a top down view on the robot. It detects and reports objects that encroach into warning and detection zones, which can be freely defined. SafetyEYE establishes whether any operators are within the action radius of the hazardous movement (safety) or have accessed a zone with an increased safety level (security).

- Projection-based workplace monitoring [40]: In this safety system developed at Fraunhofer IFF, safety regions are directly projected into an environment and cameras in the surrounding reliably detect interruptions of these projected areas. The system responds in real-time and is intrinsically safe. The safety regions can dynamically adjusted to the operation speed of a robot or other changing ambient conditions.

Concerning the use of gestures to intuitively control robots in industrial applications, several aspects have to be considered. As pointed out in [28], when human and robot are to interact directly, the interaction should take place in an appropriate way and environment, providing a certain degree of mutual understanding. For this reason, not only safety aspects have to be considered, also the human user should always be in control of the robot, especially when the robot is performing critical / potentially dangerous actions. On the other hand, the human user in this scenario should not be hindered unnecessarily to perform his own tasks, e.g., by having to carry and manipulate a device for controlling the robot while working next to it. The use of gestures in this scenario constitutes an intuitive and effortless way to control and interact with a robot. By the application of intuitive and well-known gesture sequences (i.e., mimicking human/human interaction) for commanding the robot, this way of interaction is readily available to the human user at any time (especially in critical situations), not requiring him to think about it too much and even

allowing him to perform his own tasks freely, without the need of carrying (and activating) any dedicated control device. While gesture recognition is commonly used in control interfaces or as input modality in electronic consumer devices, it is not that often used in robotic applications. Most common are gesture recognition approaches in service robotics [21]. The usage of gesture recognition in the context of industrial production is a future field of application. Many challenges must be addressed to develop robust solutions or to allow to use sensors that are better applicable under specific conditions. For example, in vision-based gesture recognition optical sensors must be developed or used that can handle differences in illumination of the environment and can be solved by appropriate methods [41]. However, other issues such as image occlusion [29] or insufficient coverage of interaction space by optical sensors are much harder to be handled in vision-based gesture recognition. Here, gesture recognition based on wearable sensors, e.g., inertial measurement unit (IMUs) that can be worn by the user [18] are advantageous. However, they also bring along some other issues such as wearability, size, weight and power supply and also drifts in the sensors that must be handled and can be more pronounced in the surrounding of robots due to the magnetic fields generated by their electric motors. Thus, holistic approaches to use gesture recognition in human-robot-interaction in the challenging environments of spacious workshop halls that can hardly be fully covered by a network of optical sensors must be developed and are currently missing. They should consider both intuitive usability and technical feasibility.

Currently, an important number of national and European projects and initiatives are trying to augment those new collaborative robots with several sensor modalities for both intuitive programming and use of the robot while simultaneously ensuring safe workspaces. For instance, the European project Saphari [2] addressed many of the essential aspects of a safe and intuitive interaction: namely, human-aware planning, reactive collision avoidance, learning and inferring motion from human behaviour, safety control aspects and compliant mechatronic designs. The project showed outstanding results in those single areas, especially in terms of safety aspects, with different robotic platforms and

demonstrations scenarios, but the goal was not to integrate all capabilities in a single platform. Also with a main focus on safety, the European project VALERI [5] proposed the development of a safe mobile manipulator for assisting human workers in aerospace production tasks. The safety of the collaboration was achieved by combining a visual workspace monitoring for tool safeguarding using a combination of a ToF (Time of Flight) camera and three pairs of stereo cameras mounted on a pan-tilt unit. Additionally, tactile sensors for safety and haptic interaction were used around the mobile platform. Another concept for integrating autonomous mobile manipulators in industrial environments is the mobile manipulator "Little Helper" [23]. The robotic system makes use of a generic product platform architecture and universal communication language as well as commercial off-the-shelf hardware components and software solutions. The robot was used in the European Project TAPAS [3] and was meant to work with or alongside people. The safety features decouple the mobile and manipulator systems: while the robot is driving, only the mobile platform is active and uses laser scanners to avoid collisions. While the mobile platform is stationary, the active use of sensors to monitor the manipulator workspace is used to reduce speed or stop the robot. In the area of intuitive interaction, the project AMIKA [6], funded by the German Federal Ministry of Research and Education (BMBF), aims at facilitating the use and programming of complex industrial machines (including, but not only, industrial robots). The idea is to take into account the specific capabilities of the particular user (age, qualification, sociocultural background) for the definition of the user interface with the machine. Also, the integration of additional sensors should help identify the needs and intentions of the user ending up with an intuitive use and programming of the machine. Finally, the European Project ROSETTA [1] also develops methods for intuitive and natural interaction with robots, having as goal intuitive and efficient programming to adapt robots to changes in the products to manufacture. This is done by representing and sharing common knowledge between robots and by learning of skills with the aim of embedding robots with more autonomy and flexibility. In terms of safety, the project uses the previously

mentioned intrinsically safe dual-arm robot Yumi from ABB.

3. System Description

The developed robotic system is based on two KUKA iiwa lightweight robots [7] equipped with 3-finger grippers from Robotiq [9] (see Fig. 1). Moreover, three RGB-D cameras (ASUS Xtion Pro Live) monitor the common human-robot shared workspace to ensure real-time collision-free robot movements. For monitoring the surroundings of the system, two SICK LMS100 laser scanners are used, which are mounted on opposite corners of the table and jointly perceive a 360° view of the area around the robot.

The developed control software is based on the DFKI's software framework Rock [15] which integrates the multiple software components and allows a fast reconfiguration of the task to automate.

The KUKA iiwa is an industrial manipulator which is commonly controlled via the KUKA Sunrise Cabinet, a control PC with a native software interface. One of the main goals of this work was the development of hardware-independent approaches for human-robot collaboration. Thus, the main control software is running on an external PC and connected to the KUKA Sunrise Cabinet using a UDP communication protocol. As a consequence, the developed robot control software is independent of the robotic hardware and could be easily transferred to other systems. On the other side, all sensors are connected to a PC exclusively dedicated to sensor processing.

The robotic system possesses three working modes:

Automatic mode: There are no humans detected in the shared human-robot workspace or approaching it. The robot executes an automatic task at the desired operating speed.

Approaching mode: The robot detects the intention of humans to enter the robot's workspace based on the information from the laser scanners; in such a case, the robot goes on with its automatic task but at reduced

Figure 1: Robotic system developed in this work.

speed as long as the human is around. In this mode, the robot is ready to receive commands via gestures; for instance, to stop its motion or to switch to Interaction Mode.

Interaction mode: The robot is set in compliant mode to allow a direct physical contact with humans (e.g., to inspect the part being held by the robot).

The switch between these modes occurs automatically by processing the real-time sensor data and/or the gestures performed by the operator. Those transitions are controlled by the Component Network Supervision (See Section 4.2) which guarantees smooth transitions by allowing transitions only after some prerequisites are met and once the system is ready to switch.

In summary, the collaborative skills of the robot are based on the seamless integration of:

- Novel multimodal sensor-based person tracking using RGB-D and laser scanner data
- Real-time identification and tracking of point clouds in the workspace to be used for collision avoidance
- Real-time dual-arm self-collision avoidance and dynamic collision avoidance with external objects
- Robot's speed and compliance automatically adjusted depending on the current working mode and real-time environment data
- Intention recognition and recognition of simple human gestures

A whole-body control framework is used as a key control element whose controller-constraints combinations constitute the basic building blocks that describe actions of a high-level action plan to be sequentially executed. The whole-body control receives inputs from all simulatenously-running controllers and generates a single output to the robot that takes into account all competing controller inputs, thus ensuring not only the optimal use of the degrees of freedom available at the robot, but also the stability of the whole control system. Additionally, a modular, robot-agnostic software control framework is used to bind all components together and allow reusing generic software components to describe a variety of complex manipulation behaviours, whilst at the same time keeping independence from the particular robot hardware [20].

4. Software Architecture

4.1. Rock Framework

The software for this work was developed using the Robot Construction Kit (Rock) [15]. Rock is a framework to develop software for robotic systems based on the component model of the Orocos Real Time Toolkit [37]. In addition to the software component model, further tools for robot software development are included. In this subsection, the relevant parts of the framework will be introduced.

4.1.1. Component models

Within Rock, components are defined by a data flow and a configuration interface as well as a common life-cycle. The life-cycle describes the transitions of states a component can go through. There are inoperative states *stopped*, *pre-operational* or *configured*. Additionally, each component has a running state and, in case of a runtime error, an error state. At runtime, a single component should only have a single well-defined operative mode, and not switch its behaviour in different runtime modes. This makes the behaviour of a component solely dependent on its configuration and the input data, but not on its history – a property that increases the possibility of re-using single components and integrating them into higher control levels [26].

4.1.2. Plan management

The plan manager *Roby* [25] included in Rock allows representing, executing and also adapting plans. An activity within a plan is defined by a graph of task relations. Task in this context means an operation of the system which is implemented by a block of user code. The relations between the tasks represent semantic dependencies between tasks, which are used to express why a task is within the plan. Progress of tasks is modeled using events representing an identifiable situation that occurred during task execution. Events appear in two types: *controllable* and *contingent*. Controllable events can be triggered by the plan manager itself by calling a procedure on the task that deterministically brings the task in the situation that emits the event. Contingent events are non-controllable and thus are triggered by other processes but the plan manager, i.e., usually the task's operation. Long-term system behaviour can be created by linking emitted events to the execution of controllable events.

4.1.3. Modeling component network models

Abstract descriptions of component networks can be created with the tool Syskit [24]. Syskit provides a Ruby-based eDSL (embedded Domain Specific Language) to describe compositions of

a) actually implemented software components,

b) an abstract operation, or

c) another component network already modeled with Syskit,

in order to model abstract functional component networks. The abstract operations (called *data services*) are defined by a minimal data flow interface and a semantic label representing their functionality. Using data services, a taxonomy-like classification processing units could be created and directly be bound to existing software components via the abstract data flow interface. To construct a fully functional component network from an abstract one, an instantiation specification must be created by selecting actual components or subnetworks as replacement for the abstract data services used in the compositions. During the instantiation of a component network description, redundancies are merged, which might occur from using identical components in multiple subnetworks or from already running components. This allows the transition of an arbitrary running component network to a new one, what leads to the ability to embed different component network descriptions into a higher-level coordination. Syskit is integrated with Roby such that Syskit's compositions can be used as Roby tasks.

4.2. Software Components

The software architecture used in this work can be divided into three main blocks: a) the Software Component Network b) the Component Network Supervision c) the Application Logic (see Fig. 2).

The Software Component Network is composed of Rock components using Syskit and the extensions that were introduced in [42]. In the Software Component Network, sensor processing or control algorithms are arranged into a topology that creates the controller loop that generates the control commands for the robots. A schematic overview of the Software Component Network is given in Fig. 2 (top). Sensor data is fed into the sensor processing pipelines that

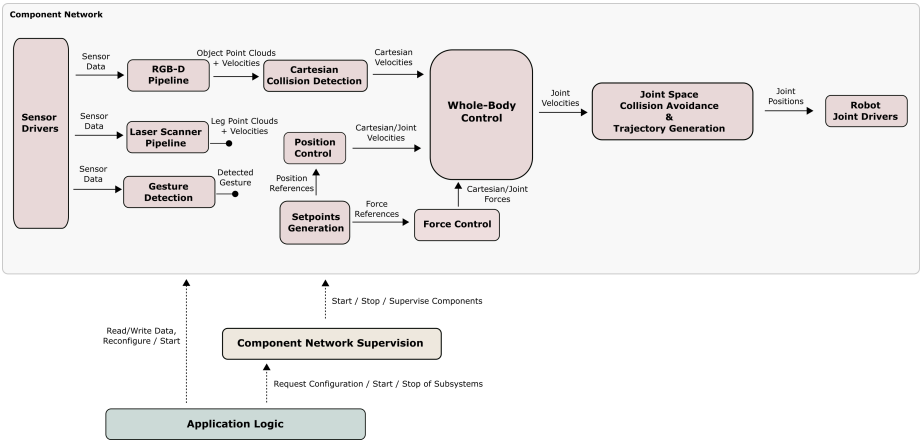


Figure 2: Overview of the main software components.

are used for detecting external objects in the environment (see Section 6 for details). The resulting object point clouds are processed by a potential field-based collision avoidance controller (see Section 7.2 for details).

The robot manipulation tasks are described as a number of joint waypoints or object-relative Cartesian waypoints. The corresponding components are referred to as *Setpoints Generation* in Fig. 2. A Whole-Body Control (WBC) component weights and prioritizes the different controller inputs and, taking into account a set of constraints, outputs the most optimal joint velocity commands (see Section 5.2 for details).

A further control layer is included between the WBC component and the robot drivers, in which both self and external collisions are ascertained in joint Space. Moreover, a trajectory generation component, based on the Reflexxes library[30], generates smooth and synchronized motion commands for the two robot arms.

The laser scanner pipeline – detecting whether someone entered the workspace (details can be found in Section 6) – as well as the gesture detection module (cf. Section 8) are also part of the Software Component Network, but have no direct data flow connection to other Rock control components. The data gen-

erated by these components is interpreted by the Application Logic which will be described in the next subsection.

Finally, a Component Network Supervision module takes care of launching the corresponding processes of the components and supervise their execution. In this way, if any component fails, it will be detected and the robot control will be immediately stopped.

4.3. High-level Action Plans

The Roby plan manager was used in this work for the application development. Here individual *tasks* can be described that might emit certain events during their execution. These tasks are integrated into a plan by binding the execution of tasks to the emission of events. This event-reactor pattern allows parallel execution of multiple tasks at the same time. For a simpler application development, high-level commands were created which allow specifying distinct temporal execution patterns (synchronization templates).

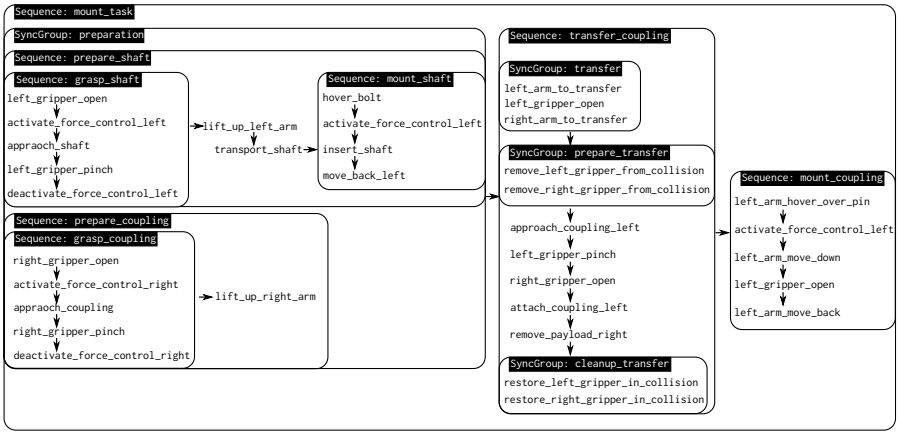


Figure 3: Temporal relation of a mounting task. The use of synchronization templates is depicted by rounded rectangles with black header. Arrows indicate the temporal order in which tasks are executed within a *Sequence*.

Figure 3 depicts the task of inserting a coupling into a gear shaft. The

synchronization patterns used here are *Sequence* and *SyncGroup*. A Sequence forwards the success of a task as the start signal of the following task. A *SyncGroup* waits until all tasks within the group have finished successfully until the success-signal of the group will be emitted.

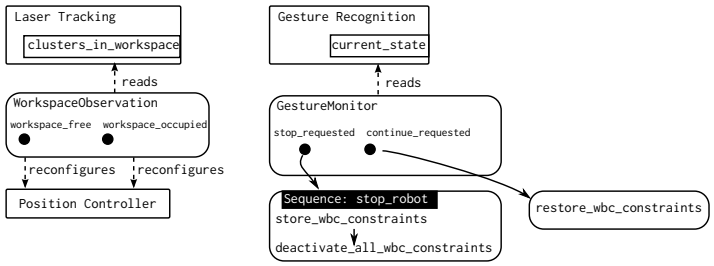


Figure 4: Tasks (rounded rectangles) can have access to components from the Software Component Network. Reconfiguration of the position controller of the arms is performed in this way by the *WorkspaceObservation* task. Custom events of tasks are depicted as black dots. Arrows indicate event forwarding. If a gesture is recognized by the *GestureMonitor* task, the corresponding event is forwarded to a stop/resume task.

Actions have access to the Software Component Network via the Component Network Supervision. Thus, here data from the Software Component Network could be analyzed within a task and be forwarded to an event that could trigger some behaviour in the high-level action plan. For example, Fig. 4 shows that the *WorkspaceObservation* task reads data from a port of the laser tracking component and, based on that, reconfigures the position controllers to move at lower speed (if the workspace is occupied) or at higher speed (if the workspace is free).

Figure 4 also shows how the *GestureMonitor* task only emits an event when a gesture state change was requested from the gesture recognition component. Within the high-level action plan, the emission of the events is then forwarded to other tasks. In this particular example, the request from a stop gesture will cause storing the current WBC task configuration and, subsequently, disabling all WBC constraints to stop the robot movement. If a *forward-gesture* is used (details in Section 8) and the robotic activity should be resumed, the previously

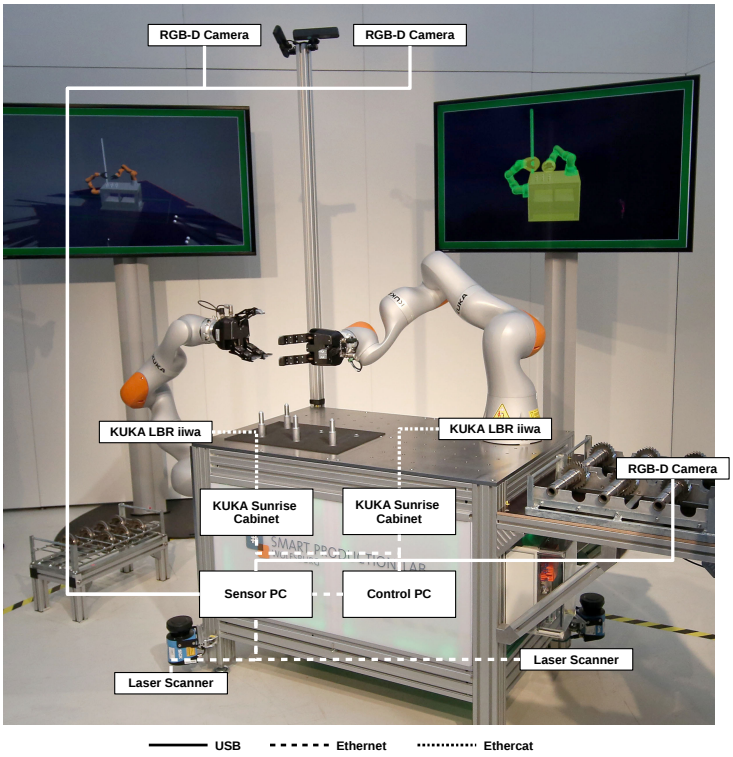


Figure 5: Hardware setup and communication between hardware components.

stored WBC constraints are restored again.

5. Robot Control

5.1. Communication between hardware components

The KUKA iiwa system is an industrial robot with a native control interface that runs solely on the KUKA Sunrise cabinet. In order to connect the robot to our Rock framework, a UDP based Client/Server application (see Fig. 5.1) was implemented. This Client is a Rock component that is able to send control commands to the arm with a frequency of 5 ms, change the dynamic model of the gripper, adapt the impedance of the robot and handle error states. By connecting to our Rock framework, both arms can be synchronously controlled

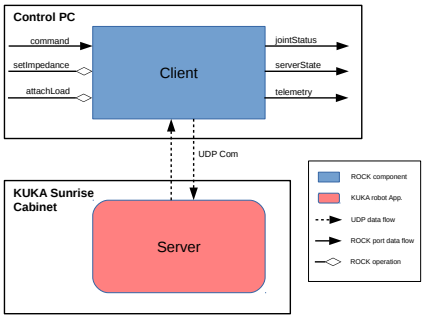


Figure 6: UDP Server/Client implementation for controlling the KUKA iiwa.

and easily integrated with other software components. The server is a Java-based robot control application that simply forwards the given commands to the robot using the DirectServo interface from KUKA.

5.2. Whole-Body Control

Robot control based on constrained optimization, also referred to as *Whole-Body Control (WBC)* [34], is a method that allows to specify and control tasks for complex robotic systems like humanoids, multi-legged walking robots or industrial dual-arm systems. The fundamental principle of WBC is to break down the control problem into several subtasks, for example "maintain an upright body posture", "apply a certain force on an object" or "keep the orientation of the gripper". Next, each subtask is described as a *constraint* to an optimization problem:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & f(\mathbf{u}) \\ \text{subject to} \quad & g_i(\mathbf{u}, \mathbf{x}) \leq 0 \quad i = 1 \dots k \\ & h_j(\mathbf{u}, \mathbf{x}) = 0 \quad j = 1 \dots l \end{aligned}$$

where f is an arbitrary goal function. The optimization problem can be subject to a number of k inequality and l equality constraints $g_i(\mathbf{u}, \mathbf{x})$ and $h_j(\mathbf{u}, \mathbf{x})$, which are a mathematical representation of the robot's task. The optimization

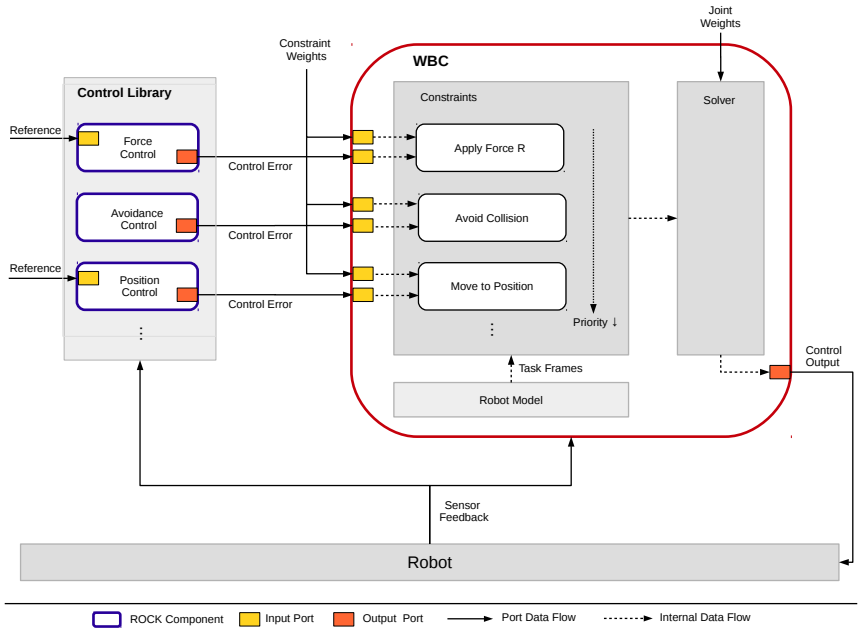


Figure 7: Overview of the Whole Body Control module.

problem is solved online and the solution $\mathbf{u}$ is applied to the actuators of the robot. In the next control cycle, the constraints $h_j(\mathbf{u}, \mathbf{x}) = 0$, $g_i(\mathbf{u}, \mathbf{x}) \leq 0$ are updated with the current robot state $\mathbf{x}$ and the process is repeated. Instead of describing robot motions (e.g., by the means of a trajectory), the desired robot behaviour is specified by a set of constraints and a numerical solver deals with the movement generation. This approach is particularly appealing, because complex robot control problems can be described by combinations of simple constraints, which are easy to specify. Furthermore, the redundancy of the robot can be optimally exploited by the solver, utilizing the degrees of freedom of the whole robot body.

In this work, a variant of the algorithm described in [36] was used. The objective function $f(\mathbf{u}$ used in this case is

$$f(\mathbf{u}) = \|\mathbf{u}\| \tag{1}$$

where $\mathbf{u}$ are the joint velocities that represent the "best possible" solution of

the constrained optimization problem in the current time instant. Thus, we minimize the kinetic energy of the robot joints, a solution that can be obtained using the (weighted) generalized inverse of the robot Jacobian.

The main components of the whole body control approach are described in Fig. 7. Each subtask is described as a combination of a controller, which governs a particular aspect of the overall task (e.g., maintain a certain force on an object), and a corresponding constraint in the optimization problem. The control action is generally performed in a task related coordinate system (*task frame*), which makes the controllers independent of the robotic platform. The importance of each subtask with respect to the others can be controlled by the means of priorities, which are enforced using Nullspace projections [35]. Additionally, the contribution of each subtask can be handled by the means of task weights, while the contribution of each robot joint can be governed by the means of joint weights. The output of the whole body controller is a joint velocity signal, which is applied to the actuators of the whole robot.

Within the overall software framework, WBC is used as the key control component. Controller/constraint combinations with their parameters form basic building blocks for actions, which can be executed in sequence in a high-level action plan (see Section 4.3). The implementation of our WBC algorithm is efficient enough to perform control actions with a $1ms$ cycle time. However, due to limitations of the robotic hardware interface, we usually use a $5ms$ cycle.

6. Environment Perception

The demonstrator robot is equipped with a heterogenous set of sensors: three RGB-D cameras and two 2D laser scanners. Through a series of processing steps, point clouds corresponding to potential collision objects (such as persons, tools or carts) are extracted from the RGB-D data and tracked. These object point clouds are directly passed on to the dynamic collision avoidance module (Section 7.2), which (if necessary) rapidly executes an avoidance motion using our whole-body control framework. The laser processing pipeline, on the other

hand, uses much of the same software components as the RGB-D pipeline to track legs in the horizontal plane and detect approaching persons in order to switch between different working modes. These two pipelines are described in more detail in the following two subsections.

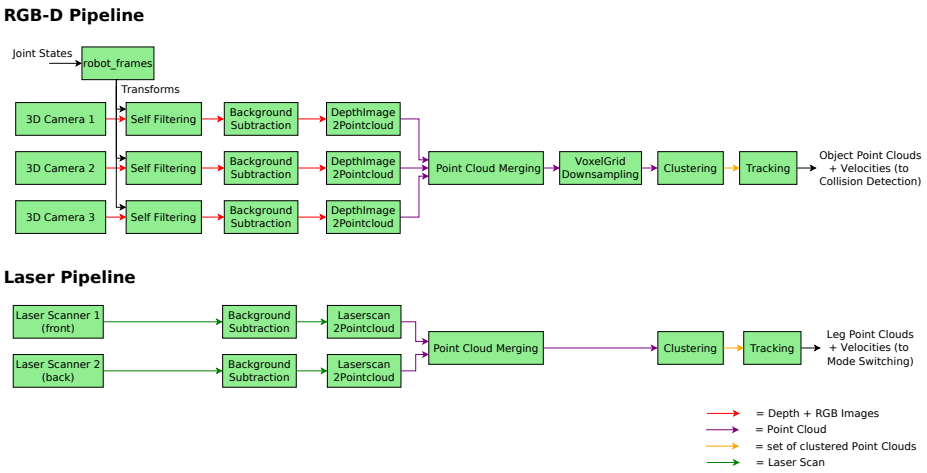
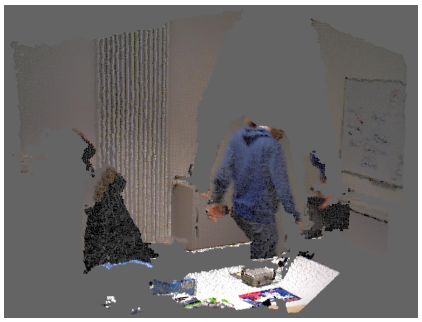


Figure 8: Overview of software components for environment perception and object tracking.

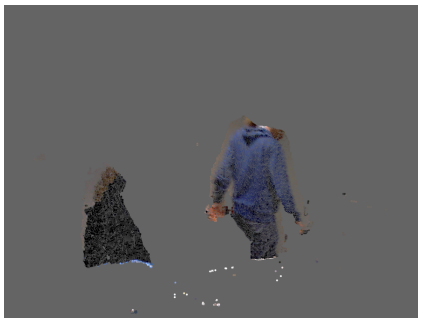
6.1. Multiple Object Tracking in RGB-D

The 3D processing pipeline is shown in the top half of Fig. 8. The initial steps work directly on the depth images of each separate camera, before the filtered depth images are converted to a fused point cloud for further processing. Some of those steps are visualized in Fig. 9.

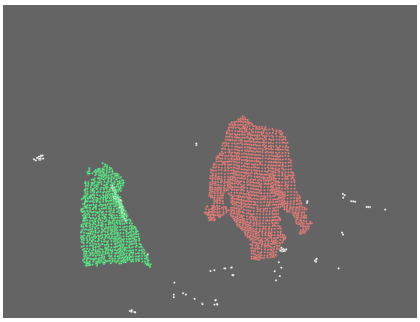
Robot self filter. The first step in the pipeline is robot self filtering, i.e., the removal of image points that lie on the surface of the robot itself. For this, the robot self filter by Blodow [19, p. 71ff] is used. It uses the URDF model of the robot and the joint angles at the time of the image acquisition to render a virtual depth image, then compares this to the actual depth image and filters out all robot points. Since this code is GPU-accelerated, it is more than fast enough to keep up with the frame rate of the cameras.



(a) full input point cloud



(b) after background subtraction



(c) after voxel grid filtering and clustering (red/green: the two object clusters; white: ignored clusters below the size threshold)

Figure 9: Point clouds at subsequent steps in the RGB-D processing pipeline.

Background subtraction. This module removes points from the depth image that belong to the static background by comparison with a reference depth image that was taken without external objects present.

Point cloud conversion and registration. The RGB and filtered depth images of all three cameras are converted to a colored 3D point cloud, transformed into a common coordinate frame and merged.

Voxel grid downsampling. The resulting merged point cloud is downsampled using a voxel grid to speed up further processing.

Clustering. The downsampled point cloud is split into clusters based on euclidean distance. Clusters that consist of fewer points than a certain threshold are considered outliers/noise and removed.

Tracking. The tracking method presented here is heavily influenced by the work of Papadourakis and Argyros [33]. A main difference is that while the method of Papadourakis and Argyros works on 2D RGB images, the proposed method processes arbitrary 3D point clouds. One key feature of both methods is that it allows objects in the input data to be under-segmented (but not over-segmented). This assumption is a good fit for the rest of the processing pipeline outlined above, since the clustering step is based on euclidean distance alone; thus, objects that are close to each other or touch each other end up in the same cluster, so the tracking method has to split them again before assigning clusters to tracks.

The tracking method models objects as 3D ellipsoids $e = (\boldsymbol{v}, \boldsymbol{R}, a, b, c)$, where $\boldsymbol{v}$ is the centroid, $\boldsymbol{R}$ is the orientation expressed as a 3×3 rotation matrix and a, b and c are the semi-axis lengths. Given a point cloud P of the object, $\boldsymbol{v}$ is equal to P's centroid. The orientation is derived by calculating the eigenvectors $\boldsymbol{e_0}$, $\boldsymbol{e_1}$ and $\boldsymbol{e_2}$ of P's 3×3 covariance matrix $\boldsymbol{\Sigma}$ and setting $\boldsymbol{R} = [\boldsymbol{e_0} \;\; \boldsymbol{e_1} \;\; (\boldsymbol{e_0} \times \boldsymbol{e_1})]$. The semi-axes can be computed as $a = \sqrt{\chi^2 \lambda_0}$, $b = \sqrt{\chi^2 \lambda_1}$ and $c = \sqrt{\chi^2 \lambda_2}$, where $\lambda_0, \lambda_1, \lambda_2$ are the eigenvalues of $\boldsymbol{\Sigma}$, and χ^2 is the critical value of the χ^2 distribution. The value of χ^2 controls the size of the ellipsoid and can be

computed from the dimensionality of the data ν and the desired ratio r of object points that are enclosed by the ellipsoid. Here, $\nu = 2, r = 0.99$ were chosen for the 2D laser data and $\nu = 3, r = 0.25$ for the RGB-D data. In other words, the ellipsoid will contain 99% of P's points on average for the 2D data (resp. 25% for the 3D data). The size of the ellipsoid controls the assignment of candidate points to tracks (see below); a larger ellipsoid will assign candidate points to a track that are further away from the track's centroid, which is advantageous when the tracked objects are moving fast (such as the tracked legs in the 2D data).

It is useful to define the distance $D(p, e)$ of a point p to an existing track's ellipsoid e in a way that takes the ellipsoid's shape into consideration, similar to what Argyros and Lourakis [17] do in the 2D case. Intuitively, a transformation $T(e)$ is computed that transforms the ellipsoid to the unit sphere, and that transformation is then applied to p. If $D(p, e) = 1$, the point p lies exactly on e; if $D(p, e) < 1$, p is inside e and otherwise outside. The affine transformation $T(e)$ can be computed from the ellipsoid $e = (\boldsymbol{v}, \boldsymbol{R}, a, b, c)$ as follows:

$$T(e) = \begin{bmatrix} 1/a & 0 & 0 & 0 \\ 0 & 1/b & 0 & 0 \\ 0 & 0 & 1/c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{R}^\mathsf{T} & -1\boldsymbol{R}^\mathsf{T}\boldsymbol{v} \\ 0 \dots 0 & 1 \end{bmatrix} \tag{2}$$

This transformation is used to define the distance measure $D(p, e)$ as follows:

$$D(p, e) = |T(e) \cdot p| \tag{3}$$

The tracking algorithm itself is shown in Alg. 1. Given a set of point clouds (one for each point cluster extracted by the processing pipeline) and a set of existing tracks, it computes an updated set of tracks. In the first step, the states (poses and velocity vectors) of all existing tracks are progressed using a Kalman filter. Next, each existing track is assigned to at most one cluster. This is done by computing the association degree $a(c, e)$ between the track's ellipsoid e and each cluster in a specified window from the track's position, and selecting the

cluster with the highest association degree (if there are any clusters inside the window). The association degree is defined as follows:

$$a(c, e) = \sum_{p \in c} e^{-D(p,e)^2} \tag{4}$$

This step results in an assignment of clusters to tracks, where a cluster can be assigned to zero, one or multiple tracks. If a cluster was not assigned to a track, a new track is created. If a cluster was assigned to exactly one track, that track is updated with the cluster data (i.e., the track's Kalman filter is updated with the cluster's centroid, and the track's ellipsoid is recomputed from the cluster points). If a cluster was assigned to multiple tracks, the cluster's points have to be split among the competing tracks. This is done by creating a set of subclusters $\{c_1, c_2, \ldots, c_n\}$ (one for each competing track) and putting each point into the subcluster of the track that has the minimum distance to the point. Formally, each point $p \in c$ is put into the subcluster c_i, where $i = \arg\max_{j \in \{1,\ldots,n\}} D(p, e_j)$. Then, each competing track is updated with the respective subcluster like in the single track case. Finally, all tracks that have not been assigned a cluster in this iteration, but were last updated within the last u_{max} iterations, are added to the result set. All other tracks are removed.

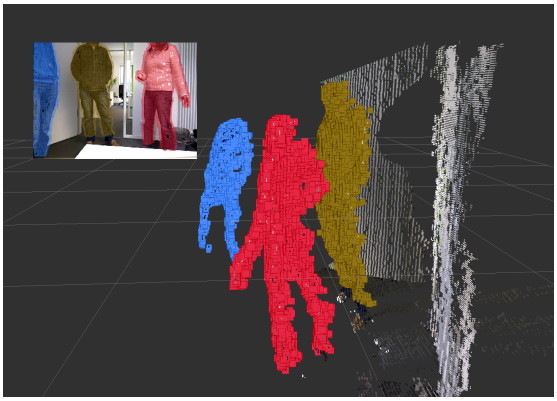


Figure 10: Three persons being segmented from the background and tracked.

One example result of the presented tracking method is shown in Fig. 10.

Algorithm 1 Tracking

Input: a set of cluster point clouds C, a set of existing tracks T

Output: a set of updated tracks T'

1: **function** TRACKING(C, T)
2: $T' \leftarrow \varnothing$
3: $T_k \leftarrow$ KALMANPREDICT(T)
4: $CT \leftarrow$ ASSIGNTRACKSTOCLUSTERS(C, T_k)
5: **for each** $\langle c, T_c \rangle \in CT$ **do**
6: **if** $T_c = \varnothing$ **then**
7: $T' \leftarrow T' \cup$ CREATETRACK(c)
8: **else if** $T_c = \{t\}$ **then**
9: $T' \leftarrow T' \cup$ UPDATETRACK(c, t)
10: **else if** $T_c = \{t_1, t_2, \ldots, t_n\}$ **then**
11: $\{c_1, c_2, \ldots, c_n\} \leftarrow$ SPLITCLUSTER(c, T_c)
12: **for** $1 \leq i \leq n$ **do**
13: $T' \leftarrow T' \cup$ UPDATETRACK(c_i, t_i)
14: **end for**
15: **end if**
16: **end for**
17: **for each** $t \in T$ **with** $\langle \_, \{..., t, ...\} \rangle \notin CT \wedge$ LASTUPDATE(t) $< u_{max}$ **do**
18: $T' \leftarrow T' \cup \{t\}$
19: **end for**
20: **return** T'
21: **end function**

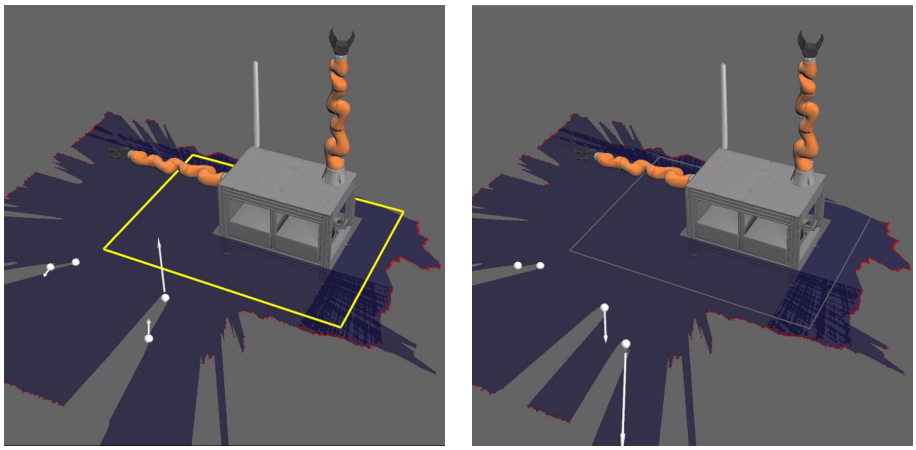


Figure 11: Workspace monitoring using laser scanner based people tracking. White balls/arrows: centroids/linear velocities of the leg tracks (1 m arrow length = 1 m/s linear velocity); yellow/grey rectangle: occupied/free robot workspace polygon; red dots: laser scanner measurements; not shown: angular track velocities, projected track trajectory). *Left:* The legs of a person move towards the workspace; the robot's speed is reduced (indicated by the yellow workspace border). *Right:* A person moves away from the workspace; the robot goes back to normal operating speed (indicated by the grey workspace border).

The point clouds of each track and their velocities are passed on to the external collision avoidance module (see Section 7.2). In the context of the full integrated system presented in this paper, the tracking method is only required to split undersegmented object point clouds and compute velocities for each object via Kalman filtering. The KCCD external object collision avoidance (and therefore the WBC controller) directly uses the full object point clouds along with these velocity vectors to predict and avoid collisions.

6.2. 2D Leg Tracking and Intention Recognition

The advantage of the laser scanners is that they cover a horizontal plane for the full 360° around the robot, avoiding the blind spots in the RGB-D image data. The laser scanners are used to track the legs of people and switch between the three working modes (automatic mode, approaching mode and interaction mode; see Section 3).

The 2D processing pipeline mostly consists of the same steps as the 3D pipeline (compare Fig. 8). The only differences are:

- no self filter is necessary, since the robot never reaches into the laser scanner's plane;
- a laser scan based (instead of depth image based) background subtraction is used; and
- no voxel grid downsampling is necessary, since the number of points is comparatively low.

The positions and velocities of the tracked legs (as estimated by the Kalman filter) are shown in Fig. 11. They are fed to the *WorkspaceObservation* task (Fig. 4). This task projects the leg trajectory one second into the future, based on its linear and angular velocities and a linear motion model. When a trajectory intersects the workspace polygon, the system is switched into approaching mode, and the *WorkspaceObservation* task reconfigures the position controllers so that the robot moves at reduced speed. When the workspace is free again, the system goes back to automatic mode (normal speed).

7. Safety Aspects

According to the ISO 10218 standard and the new ISO/TS 15066 Technical Specification, there a currently four modes allowed for human-robot collaboration:

1. Safety-rated monitored stop, in which there is a separation between human and robot and no robot motion is allowed when the operator is in the collaborative workspace,
2. Hand guiding, in which the robot motion is only happening through direct input of the operator (by, for instance, holding a device on the end-effector of the robot),

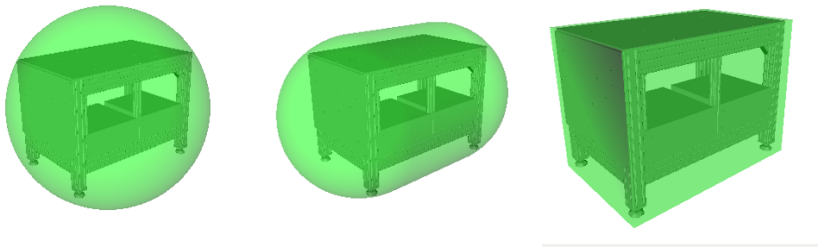
3. Speed and separation monitoring, in which there is an external sensor (usually camera or laser scanner) ensuring that there is a minimum separation between human and robot at all times. The human can work closely to the robot and without a fence, but if a minimum distance is reached, the robot stops,

4. Power and force limiting by inherent design or by control, in which the maximum forces (or power) that the robot can exert are limited by either mechatronic design or by control software. This is the only mode in which physical contact between human and a moving robot is permitted.

The robotic system presented in this paper addresses the third and fourth cases. The system offers speed and separation monitoring in the sense that external sensors are permanently monitoring the environment to avoid collisions between humans and robot. However, in our current setup, the robot is not stopped when a human is detected but the robots react in two ways: first, when the laser scanners detect the access of a human into the workspace, the robots automatically reduce their working speed; secondly, by using real-time data from the 3D cameras, the robots will try to find escape trajectories to dynamically avoid colliding with any object in the environment (including possible humans). Likewise, the system satisfies the force limiting control case by automatically changing to compliant mode once the operator requests an interaction.

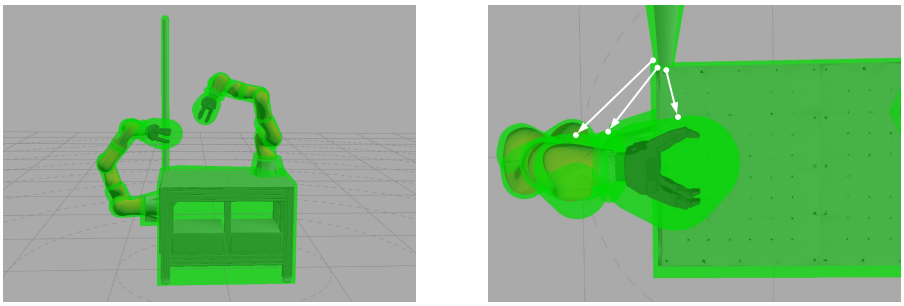
7.1. Robot Self-Collision Avoidance

Our approach for self-collision avoidance is based on the KCCD library [38], which contains methods to rapidly evaluate distances between links of a robotic manipulator and compute ideal brake timing for its actuators. In KCCD, a rigid body is described by the means of a convex hull enclosing a finite set of points p_i, $i = 1, \ldots, n$, which is extended by a buffer radius r (see Fig. 12). The complete collision model of the dual arm system can be seen in Fig. 13(a). If the robot is moving, the collision volumes grow and shrink according to a braking model that includes the current moving velocity and desired deceleration of each



(a) $n = 1$, $r = 0.81m$ (b) $n = 2$, $r = 0.60m$ (c) $n = 8$, $r = 0.001m$

Figure 12: Body representation in KCCD. Each body is described by a convex hull enclosing n points and a buffer radius r.



(a) Collision model (b) Repulsive vectors

Figure 13: Self-collision avoidance of the dual arm system.

joint. This way, the optimal brake timing can be determined, slowing down the manipulator smoothly rather then performing a hard stop.

7.2. Dynamic Collision Avoidance with External Objects

The KCCD library for self collision avoidance described in the previous section requires a formal model of the robot geometry and all collision bodies. Obviously, objects that enter the workspace of the robot can be of arbitrary shape and size, which makes the method infeasible for determining external collisions. Thus, the library was extended with an interface for adding arbitrary bodies to the model at runtime. Furthermore, an approach for quickly converting clustered 3D point clouds into KCCD body representations was developed.

This process works as follows:

1. Add a KCCD supporting point for every point in the point cloud
2. Iteratively remove each supporting point from the KCCD volume and check if the volume still encloses the complete point cloud
3. If the number of points n is bigger than the desired maximum number of points, increase the buffer radius r by a fixed step and go to step 2. Else, terminate.

Further, speed and robustness was improved by starting the computation with the KCCD volume from the previous cycle. The computed convex hull is not "optimal" in the sense that it represents the smallest possible volume that encloses all points of the point cloud. However, accuracy is not an issue here, since what is required is that the robot avoids obstacles that enter the workspace with a large safety distance.

Given the distance computations between the robot and external objects, avoidance motions are computed using virtual repulsive potential fields [27]. The repulsive vector is spanned between the two closest points on obstacle and manipulator, respectively (see Fig. 13(b)), pointing away from the obstacle. To execute avoidance motions, our whole-body control framework, described in Section 5.2, was used. Each relevant collision body of the manipulator is assigned an avoidance controller and a corresponding constraint with a certain priority. This way, the avoidance behaviours can be integrated with the actual task goals of the robot.

7.3. Compliant Mode

Besides controlling the joint positions of the manipulator arms it is also necessary to control the impedance of the arms and to adapt the impedance task-specific parameters from within the control framework. In automatic mode, a high stiffness may be used to move with high accuracy. On the other hand, when doing tasks involving human interaction, e.g., when allowing the inspection of

parts, the robot arm needs to be highly compliant. To provide access to these settings and control variables from the Rock control framework, two interfacing components have been developed.

One of them is implemented using the JAVA programming environment of the robot's manufacturer. It translates the robot-specific commands to and from packets communicated via Ethernet connection. This component is executed in soft real-time with a sampling time of 5 ms on each robot controller (KUKA Sunrise Cabinet, Fig. 5.1). In particular, the methods supplied by the package `com.kuka.roboticsAPI` [32] are used. Since the joint position commands received from the control framework are already interpolated, a more direct execution on the robot controller is desired. Therefore, the motion commands are forwarded to the `DirectServo` [31] interface. It allows for continuous reference updates and does not additionally constrain the commanded motion wrt. acceleration or its derivative. As control mode, `JointImpedanceControlMode` is set.

On the other side, the second component is integrated into the Rock control framework. It communicates with the robot via Ethernet, makes the status information available via ports to the other components of the control framework and forwards the commands received from them to the robot.

As a high stiffness setting, a value of 400 N m/rad for all joints is set. For low stiffness, values of 50 N m/rad for joints 1 to 4, a value of 10 N m/rad for joints 5 and 6, and a value of 5 N m/rad for the 7th joint is used. The damping value is set to 1 N m s/rad in both cases.

8. Gesture Recognition

A key aspect of future intelligent assistance systems for industrial applications is the intuitive interaction with humans. There are multiple approaches for the communication between humans and robots, but in industrial settings only a few are suitable in order to allow a safe and easy operation. Humans usually communicate using voice, which is difficult to implement for a robot in

noisy environments like in automotive assembly lines. Also the operation via input devices like keyboards or touch displays is not applicable, since workers often need two hands free for their jobs in assembly. Therefore the interaction is implemented via gestures. The challenge of using gestures for the operation of a robotic assistant system is that the recognition needs to be very robust in order for the technology to get accepted by the workers. Furthermore it can be dangerous for workers and robots if gestures are not immediately and correctly recognized or if specific movements of the worker are interpreted as gestures although this was not intended by the worker. In order to avoid such situations, three simple but unique gestures are defined that allow the intuitive operation of the whole dual arm system; the forward-gesture, the pause-gesture, and the collaboration-gesture.

The *forward-gesture* is used to move the system to the next state or to reactivate the system if its movements were paused by the operator. In order to trigger the forward-gesture the right arm is stretched and moved twice from right to left, like a wipe movement performed twice in a row. This gesture was defined because it is an intuitive way to indicate the proceeding to a next state (e.g., signaling a human coworker to "go on") and is not likely to be accidentally performed during normal operation in car assembly. The *pause-gesture* is used to pause the movement of the robot, for example to enter the workspace in a static state. It is triggered by rising the right arm above the head, which is also very intuitive to indicate a stop command, also unlikely to occur accidentally while working with the robot. These two gestures, to bring the robot to a stop and continue again, are very important to control the robot without a physical interface. Especially the *pause-gesture* can be used as an emergency stop without carrying around an explicit emergency stop switch. However, in complex environments more functionality can be needed. Therefore a third gesture was defined to activate a special mode of the robot, i.e., collaboration with the operator. This *collaboration-gesture* is triggered by stretching the left arm to the front and perform a pulling movement to the body ("over here"). By using these three gestures the whole schedule of the robot system can be

controlled in an intuitive way.

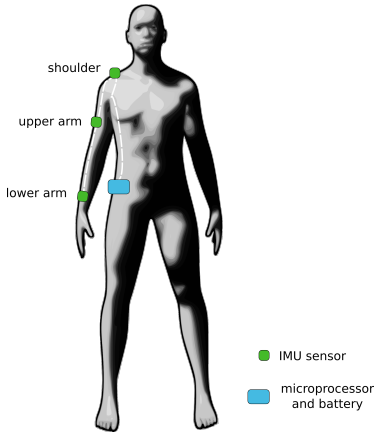


Figure 14: Placement of sensors used for gesture recognition.

The actual recognition of the gestures is performed by a dedicated hardware device integrated into the operator's cloths (cf. Fig. 14). This device consists of three main parts: 1) a string of 3 sensor modules (IMU, 9 DoF) capturing the raw movements of the operator's right arm, 2) a small microprocessor board which is computing posture information from the sensor's raw data streams and maintaining WiFi connectivity and 3) a battery pack. This setup allows for independent operation of the gesture recognition, i.e., it does not require cable connections for operation and may be applied to the operator independent of the location of the robot to be controlled. Beyond that, by application of the microprocessor board which is able to preprocess all needed posture and gesture data, very little bandwidth is required to communicate with the robot (less than 1 kB/s), allowing for robust radio communication even in noisy or crowded environments. Within the gesture recognition, gestures are defined as being *timed series of postures*, i.e., to issue a specific gesture command, the operator has to match the series of postures the gesture consists of within a given time interval. As the microprocessor board is loaded with a set of software modules, besides the core recognition of gesture commands given, it is also possible to *record* and save postures for definition of further gestures. That is, in order

to define a new gesture command, the operator simply goes into each of the desired postures this new gesture should consist of. The postures recorded this way can then be aggregated to the gesture command and an action (e.g., sending a command to the robot) can be assigned to it. This allows for very fast, flexible, and intuitive extension of the gesture recognition.

9. Practical Application

In order to underline the practical relevance as well as the future potential of this demonstrator, a representative handling and assembly scenario from a gearbox manufacturing plant is chosen. The current workplace is depicted in Fig. 15. Here, the gear shaft and a coupling have to be joined manually with a tight tolerance, which is a tiring job since the parts are heavy, the surfaces easily scratched, and the gear wheels are shock sensitive. Therefore, the chosen workplace is a good example for a manufacturing scenario where a collaborative robot would highly improve the ergonomic situation of the worker. Also, it is a technologically demanding situation because of space restrictions and the resulting danger of injuries through edgy parts being handled, which makes it necessary that the robots show compliant and sensitive behaviour.

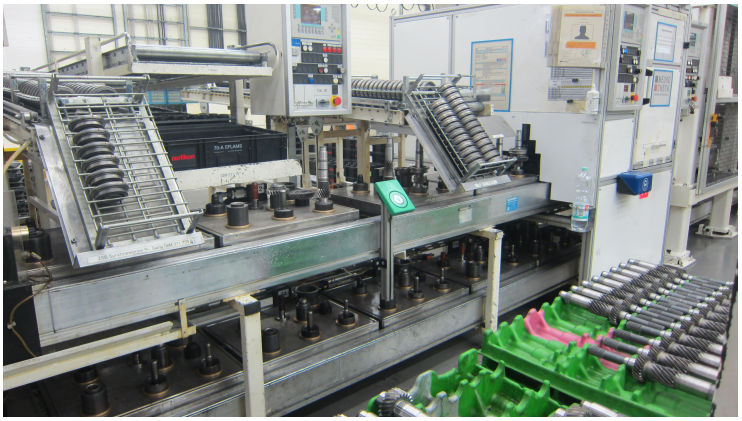


Figure 15: Pre-assembly of the gearbox with shafts and couplings at a Volkswagen plant.

For the practical demonstration of the system, the gearbox scenario is adapted

as depicted in Fig. 1. While there are some simplifications regarding the environment of the workplace, the main tasks to be performed by the robots (handling and precise assembly) are executed. Further, the setup with the two arms being mounted in different orientations is chosen to achieve an adequate spatial coverage of the robot system. In combination with the safety and interaction aspects discussed above, the demonstrator therefore addresses a solid cross section of tasks and technological challenges, which can be transferred to other applications within the Volkswagen group. To underline the feasibility of human-robot collaboration, the standardized work description of the given workplace is also analyzed using the method shown in [39]. This is done by calculating the weighted sum of an estimated automation potential for each task category and duration. The results of the human-robot potential analysis are illustrated in Fig. 16. The diagram shows the automation potential of all tasks performed by the human worker and the duration normalized to the length of one assembly cycle. Time is encoded on the angular axis and the estimated potential is drawn on the radial axis. By calculating the weighted sum of all subtasks an overall automation potential of 32% is achieved. This value is a good base for the deployment of a robotic assistant since the robot can be utilized to support tedious and exhausting subtasks with a high automation potential, e.g., pick & place of heavy parts.

The procedure of the implemented scenario starts in the initial state shown in Fig. 17(a). By executing the *forward-gesture* in Fig. 17(b), the system is switched to a mode in which its self-collision avoidance is demonstrated. During that mode the robots avoid collision with static as well as with dynamic objects in the environment (Fig. 17(c,d)). At any time during the process, the system can be paused by the operator by executing the *pause-gesture*, shown in Fig. 17(e). When paused, the robots are switched to compliant mode and can be safely manipulated by the operator (Fig. 17(f)). When receiving the *forward-gesture*, the assembly is resumed (Fig. 17(g)). The arm located on the top of the table is responsible for most of the assembly steps and for the interaction with the worker. Because of its position near the assembly area on the table,

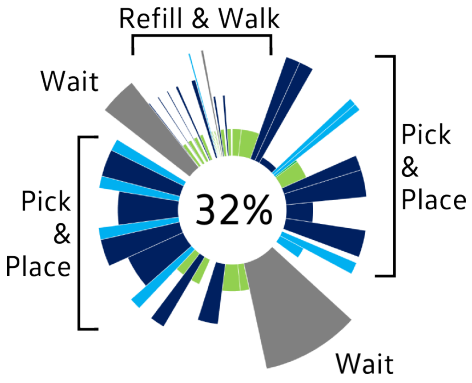


Figure 16: Visualization of normalized task duration (angular axis) and the estimated automation potential (radial axis) of the specific workplace within the gearbox assembly line. The type of subtask is encoded in the color: pick/pick&place (blue/light blue), walk (green) and wait/process time (grey). Total automation potential calculated by the weighted sum of all subtasks is 32%.

this arm is able to perform actions in almost every orientation, which is important for the transfer to other applications. In this specific scenario, the arm grasps the gear shaft from the container mounted at the right side of the table and puts these onto the bolts on the left side of the table using force feedback (shown in Fig. 17(h,i)). The arm mounted on the side of the table uses a pinch movement to grasp a coupling from the storage located on the floor (Fig. 17(j)). The coupling is then moved to a transfer position above the table where the workspaces of both robots overlap. Since only the top mounted arm can reach every bolt position on the table, the coupling is exchanged between the two arms (shown in Fig. 17(k,l)). The top mounted arm then executes the final assembly step by inserting the coupling into the gear shaft (Fig. 17(m)). This final step is executed using force control because the fit between both parts has low tolerances and can easily tilt. During the whole operation, the robots' speed is adapted depending on the distance to the operator. The current working mode is indicated by LEDs on the front of the system and on a monitoring screen (Fig. 17(n,o)). By executing the *collaboration-gesture*, shown in Fig. 17(p), the worker can request the collaboration mode. In this mode, the robot then grasps

the last assembled piece and suspends it in a comfortable position for the operator to inspect it (Fig. 17(q,r)). The whole process is repeated again for every gear shaft [1].

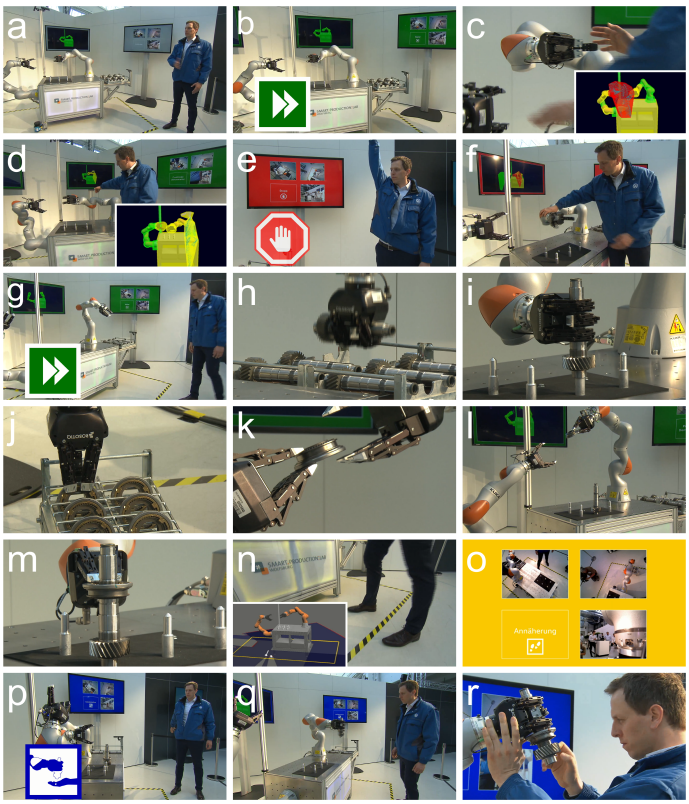


Figure 17: Timeline of the practical application. It shows the self-collision avoidance (a–d), pause mode (e,f), the assembly process (g–m), adaption of the working mode by workspace monitoring (n,o), and the collaboration mode (p–r).

10. Conclusions

Human-robot interaction requires a holistic approach that coherently integrates different subcomponents into a more complex system. Currently, great

[1] see complete video under `https://youtu.be/VoU3NbTyFtU`

advances are seen in different areas of robotics: from hardware and mechatronic design, from the perception, control and autonomy algorithms, and up to the planning, learning and reasoning capabilities. However, one of the remaining challenges is currently the integration of such capabilities into a single system which possesses greater abilities than merely the sum of its single subcomponents. As it has been shown, the current work aimed at integrating algorithms from different areas to create an innovative robotic system for a safe and intuitive human-robot collaboration. The solution relied on four key components: firstly, on a multisensor-based approach and tracking algorithms which supplied real-time annotated data about: (a) the status of the shared human-robot workspace (through RGB-D cameras and laser scanners), and (b) the human commands (through an IMU-based gesture recognition system). Secondly, real-time reactive collision avoidance algorithms for ensuring collision-free robot movements (both self collisions and collisions with external obstacles). Thirdly, a whole-body control component which receives simultaneously-running, and usually competing and conflicting, robot controllers' outputs and generates optimal actuator signals for the robot whilst taking into account a number of robot constraints. Finally, the deployment, control, and online reconfiguration and monitoring of such a complex network of software components is accomplished by using a powerful, modular and reusable robot-agnostic software framework and its accompanying development tools.

11. Outlook

The test results of the presented technological demonstrator are promising and at the same time outline the next steps for the industrialization of the system and its components: (1) The safety measures, especially the collision avoidance system, which marks an important improvement of collaborative robots, need to be certified for industrial use. In this case, work on two areas is required: on the one side, safety-rated versions of sensors like the RGB-D cameras are needed; on

the other side, the software needs to comply with the current safety standards[2]. (2) The capabilities for programming and control of robots independent from any proprietary software coming from robot manufacturers should be expanded and further developed. Also approaches for the intelligent configuration and subsystem integration (e.g., sensors, grippers) as well as automated programming, based on information from the production database, are needed to achieve the necessary flexibility of future robotic assistant systems. This would facilitate the integration of new intelligent robotic systems not only in the automotive sector, but in manufacturing industry in general. (3) The hardware and software architecture of the system should be optimized with regard to operating speed. Many possible applications for human-robot-collaboration need to comply with a given cycle time in order to satisfy existing or planned work processes and in consequence economic efficiency. (4) Intuitive interaction concepts between human and robot need to be further researched and developed; the future seamless collaboration between human and robot in dynamical industrial environments will not only be based on robust safety measures and intelligent control but also on improved interaction methods and intention recognition.

References

References

[1] European Project ROSETTA. `http://fp7rosetta.org/`, 2012. [Online; accessed 16-August-2016].

[2] European Project SAPHARI. `http://www.saphari.eu/`, 2012. [Online; accessed 16-August-2016].

[3] European Project TAPAS. `http://www.tapas-project.eu/`, 2014. [Online; accessed 16-August-2016].

[2] A previous version of the collision avoidance software used in the project was certified according to IEC 61508-3 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems) up to SIL (Safety Integrity Level) 3

[4] SafetyEye from PILZ. http://brochures.pilz.nl/bro_pdf/SafetyEYE_2014.pdf, 2014. [Online; accessed 16-August-2016].

[5] European Project VALERI. http://www.valeri-project.eu/, 2015. [Online; accessed 16-August-2016].

[6] AMIKA Project *(in German)*. http://rob.ipr.kit.edu/english/308_2042.php, 2016. [Online; accessed 16-August-2016].

[7] KUKA LBR iiwa. www.kuka-lbr-iiwa.com, 2016. [Online; accessed 16-August-2016].

[8] Rethink Robotics. www.rethinkrobotics.com, 2016. [Online; accessed 16-August-2016].

[9] Robotiq. http://robotiq.com/, 2016. [Online; accessed 16-August-2016].

[10] Universal Robots. www.universal-robots.com, 2016. [Online; accessed 16-August-2016].

[11] AURA collaborative robot from COMAU. http://www.comau.com/EN/media/news/2016/06/comau-at-automatica-2016, 2016. [Online; accessed 16-August-2016].

[12] CR-35iA collaborative robot from Fanuc. http://www.fanuc.eu/pt/en/robots/robot-filter-page/collaborative-cr35ia, 2016. [Online; accessed 16-August-2016].

[13] Franka robot from KBee AG . https://www.franka.de/, 2016. [Online; accessed 16-August-2016].

[14] Motoman HC10 collaborative robot from Yaskawa. https://www.yaskawa.eu.com/en/news-events/news/article/news/motoman-hc10-collaborative-robot-safe-and-flexible-interaction/, 2016. [Online; accessed 16-August-2016].

[15] ROCK, the Robot Construction Kit. http://rock-robotics.org/stable/, 2016. [Online; accessed 16-August-2016].

[16] YuMi collaborative dual-arm robot from ABB. http://new.abb.com/products/robotics/yumi, 2016. [Online; accessed 16-August-2016].

[17] Antonis A. Argyros and Manolis I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In Tomás Pajdla and Jiri Matas, editors, *Computer Vision - ECCV 2004, 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part III*, volume 3023 of *Lecture Notes in Computer Science*, pages 368–379. Springer, 2004. doi: 10.1007/978-3-540-24672-5_29.

[18] Ari Y. Benbasat and Joseph A. Paradiso. An inertial measurement framework for gesture recognition and applications. In Ipke Wachsmuth and Timo Sowa, editors, *Gesture and Sign Language in Human-Computer Interaction: International Gesture Workshop, GW 2001 London, UK, April 18–20, 2001 Revised Papers*, pages 9–20. Springer, Berlin, Heidelberg, 2002. ISBN 978-3-540-47873-7.

[19] Nico Blodow. *Managing Belief States for Service Robots*. Dissertation, Technische Universität München, München, 2014.

[20] José de Gea Fernández, Dennis Mronga, Malte Wirkus, Vinzenz Bargsten, Behnam Asadi, and Frank Kirchner. Towards describing and deploying whole-body generic manipulation behaviours. In *2015 Space Robotics Symposium. Space Robotics Symposium, Present and Future Robotics in Space Applications, October 29-30, Glasgow, United Kingdom*, University of Strathclyde, 2015. IET.

[21] Michael Van den Bergh, Daniel Carton, Roderick de Nijs, Nikos Mitsou, Christian Landsiedel, Kolja Kühnlenz, Dirk Wollherr, Luc J. Van Gool, and Martin Buss. Real-time 3d hand gesture interaction with a robot for understanding directions from humans. In Henrik I. Christensen, editor, *20th IEEE International Symposium on Robot and Human Interactive*

Communication, RO-MAN 2011, Atlanta, Georgia, USA, July 31 - August 3, 2011, pages 357–362. IEEE, 2011. doi: 10.1109/ROMAN.2011.6005195. URL http://dx.doi.org/10.1109/ROMAN.2011.6005195.

[22] Gerd Hirzinger, Norbert Sporer, Alin Albu-Schäffer, Matthias Hähnle, R. Krenn, A. Pascucci, and Markus Schedl. DLR's torque-controlled light weight robot III - are we reaching the technological limits now? In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 1710–1716. IEEE, 2002. ISBN 0-7803-7273-5. doi: 10.1109/ROBOT.2002.1014788. URL http://dx.doi.org/10.1109/ROBOT.2002.1014788.

[23] Mads Hvilshoej, Simon Boegh, Ole Madsen, and Morten Kristiansen. The mobile robot Little Helper: Concepts, ideas and working principles. In *Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1–4, 2009.

[24] Sylvain Joyeux and Jan Albiez. Robot development: from components to systems. In *6th National Conference on Control Architectures of Robots*, 2011. URL http://hal.inria.fr/inria-00599679/.

[25] Sylvain Joyeux, Frank Kirchner, and Simon Lacroix. Managing plans: Integrating deliberation and reactive execution schemes. *Robotics and Autonomous Systems*, 58(9):1057–1066, September 2010. ISSN 09218890.

[26] Sylvain Joyeux, Jakob Schwendner, and Thomas M Roehr. Modular Software for an Autonomous Space Rover. In *Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)*, Montreal, Québec, Canada, 2014.

[27] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505, Mar 1985.

[28] Andrea Elsa Kirchner, Jose de Gea Fernandez, Peter Kampmann, Martin Schröer, Hendrik Jan Metzen, and Frank Kirchner. Intuitive interaction with robots – technical approaches and challenges. In Rolf Drechsler and Ulrich Kühne, editors, *Formal Modeling and Verification of Cyber-Physical Systems: 1st International Summer School on Methods and Tools for the Design of Digital Systems, Bremen, Germany, September 2015*, pages 224–248. Springer Fachmedien, Wiesbaden, 2015. ISBN 978-3-658-09994-7.

[29] Irene Kotsia, Ioan Buciu, and Ioannis Pitas. An analysis of facial expression recognition under partial facial image occlusion. *Image Vision Comput.*, 26(7):1052–1067, July 2008. ISSN 0262-8856.

[30] T. Kröger and F. M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, 26(1):94–111, Feb 2010. ISSN 1552-3098.

[31] KUKA Roboter GmbH. *KUKA Sunrise.Connectivity Servoing 1.7*, April 2015.

[32] KUKA Roboter GmbH. *KUKA Sunrise.OS 1.7, KUKA Sunrise.Workbench 1.7, Bedien- und Programmieranleitung*, April 2015.

[33] Vasilis Papadourakis and Antonis A. Argyros. Multiple objects tracking in the presence of long-term occlusions. *Comput. Vis. Image Und.*, 114(7): 835–846, 2010.

[34] Luis Sentis. *Synthesis and Control of Whole-body Behaviors in Humanoid Systems*. PhD thesis, Stanford University, Stanford, CA, USA, 2007. AAI3281945.

[35] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*, chapter 26, pages 619–622. Springer Science & Business Media, 2008.

[36] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter. iTASC: a tool for multi-sensor integration in robot manipulation. In *Multi-sensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008.*

IEEE International Conference on, pages 426–433, August 2008. doi: 10.1109/MFI.2008.4648032.

[37] Peter Soetens and Herman Bruyninckx. Realtime hybrid task-based control for robots and machine tools. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 259–264, Barcelona, Spain, 2005. ISBN 078038914X.

[38] Holger Taeubig, Berthold Baeuml, and Udo Frese. Real-time swept volume and distance computation for self collision detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, CA, USA*, 9 2011.

[39] Johannes Teiwes, Timo Bänziger, Kunz Andreas, and Wegener Konrad. Identifying the Potential of Human-Robot Collaboration in Automotive Assembly Lines using a Standardised Work Description. In *Proceedings of the 22nd International Conference on Automation and Computing (ICAC)*, Colchester, United Kingdom, (accepted) 2016. IEEE.

[40] Christian Vogel, Christoph Walter, and Norbert Elkmann. A projection-based sensor system for safe physical human-robot collaboration. In *IROS*, pages 5359–5364. IEEE, 2013.

[41] Wei Wang, Drazen Brscic, Zhiwei He, Sandra Hirche, and Kolja Kühnlenz. Real-time human body motion estimation based on multi-layer laser scans. In *8th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2011, Incheon, Korea (South), November 23-26, 2011*, pages 297–302, 2011.

[42] Malte Wirkus. Towards Robot-independent Manipulation Behavior Description. In *Proceedings of the 5th International Workshop on Domain-Specific Languages and models for ROBotic systems*, Bergamo, Italy, 2014. arxiv.org.