Noname manuscript No.
(will be inserted by the editor)

Design Principles for Safety in Human-Robot Interaction

Manuel Giuliani · Claus Lenz · Thomas Müller · Markus Rickert · Alois Knoll

the date of receipt and acceptance should be inserted later

Abstract The interaction of humans and robots has the potential to set new grounds in industrial applications as well as in service robotics because it combines the strengths of humans, such as flexibility and adaptability, and the strengths of robots, such as power and precision. However, for a successful interaction the safety of the human has to be guaranteed at all times. This goal can be reached by the use of specialised robot hardware but we argue that safety in human-robot interaction can also be done with regular industrial robots, if they are equipped with additional sensors to track the human's position and to analyse the human's verbal and non-verbal utterances, and if the software that is controlling the robot is especially designed towards safety in the interaction. For this reason, we propose three design principles for an increased safety in robot architectures and any other software component that controls a robot for human-robot interaction: *robustness*, *fast reaction time*, and *context awareness*. We present a robot architecture that is based on these principles and show approaches for speech processing, vision processing, and robot control that also follow these guidelines.

Keywords Human-Robot Interaction, Safety Issues, Speech Processing, Vision Processing, Robot Control

Technische Universität München
Robotics and Embedded Systems
Boltzmannstraße 3
85748 Garching bei München
Germany
E-mail: {giuliani,lenz,muelleth,rickert,knoll}@in.tum.de

1 Introduction

Robots are used in factories all over the world because they are fast, they can lift heavy weights, they can operate in environments that are dangerous for humans, and they can repeat the same movements over and over in a precise way. However, the application of robots has one main disadvantage: robots need to be reprogrammed for every new task they should execute. Of course, one could argue that then robots should be used for repetitive tasks that need to be executed many times, such as welding a car body, but would it not be useful to combine the obviously excellent capabilities of a robot with the flexibility and adaptability of humans? The combination of the strengths of humans and robots could potentially lead to new ways in production in which customers can choose a product that is especially customised for them but still was made with robot precision.

When humans should work together with industrial robots there is one big limitation: industrial robots are heavy machines that can move very fast and can easily hurt a human user. Haddadin et al. report in [21] the possible dangers arising for a human who is interacting with an industrial robot. They conducted a systematic series of tests with several industrial robots hitting crash test dummies and listed the injuries that might be caused by uncontrolled robot movements, which included fractures and shearing of limbs. This is the reason why in factories robots usually are set up behind fences and the only interaction of human and robot takes place when the robot is turned off for maintenance.

But how can robots be made safer for the interaction with humans? One way would be to build safe robot hardware, which was for example done for Justin, a light-weight robot especially designed for interaction with unknown environments and with humans [32]. This approach has two main disadvantages: first, robots that are especially built for a safe interaction as a trade-off often lose one or more of their abilities that makes them interesting. For example they cannot lift heavy loads any more or have to move slowly. Second, the development and money that companies invested in regular industrial robots would be lost if they had to switch to new safer robots for human-robot interaction (HRI).

Therefore, in order to make industrial robots safe for HRI, we argue that these robots need to be equipped with additional sensors that enable the robot to determine the human's position and to understand the human's utterances (which includes not only speech and gestures, but also also other cues, including affective states). Additionally, the robot needs to be controlled with a robot architecture and software that is especially tailored for safety in HRI. The safety of the human that interacts with a robot is one of the prerequisites to build social robots, since a friendly social interaction is not possible if one of the partners is a threat to the other. For this reason, we propose three elementary design principles for every software that controls a robot in a safe way: *robustness*, *fast reaction times*, and *context awareness*. These design principles are the outcome of development work in several projects for HRI (for example JAST[1] and JAHIR[2]) and have proven to be essential for any software component in this field.

The remainder of this publication is organised as follows: Section 2 introduces the three design principles for safety in HRI and shows how modules for input processing and robot control can be combined in a robot architecture that follows these principles. After that, we show how the safety design principles can be applied in approaches for speech processing (Section 3), vision processing (Section 4), and robot control (Section 5). Finally, Section 6 concludes this publication and discusses future research directions.

2 Safety Design Principles

In this section, we describe three design principles for robot architectures for safe HRI and show an example of such an architecture. Our research should extend previous work by the CoSy project [22], which developed three design principles for cognitive systems. We also regard these principles as essential for any goal-directed robotic system that shows cognitive abilities:

– **Concurrent modular processing.** Robot architectures have to consist of single modules that run in parallel. With this design, complicated tasks can be split into subtasks that are executed in parallel by specialised parts of the system, for example the vision system may track a human while the actuator control makes sure that the robot does not collide with the human. This design principle also implies that the system consists of several specialised subarchitectures, which can be renewed or extended easily due to the modular architecture design.
– **Structured management of knowledge.** This principle organises the information representation in the system and the flow of information between the single parts of the system. The information inside the architecture is defined by subarchitecture ontologies and general ontologies, were the term "ontology" is a general expression that stands for a representation format. This means that each subcomponent of the system has its own representation for its knowledge. For example, the vision system might have an internal representation for objects that consists of vectors that contain colour and shape information while externally it might translate this information into a string format for other subarchitectures.
– **Dynamic contextual processing.** This principle says that in order to implement a goal-oriented behaviour in the cognitive system there have to be control mechanisms that steer the information flow of the concurrently working system components. The CoSy project proposed to implement this behaviour in a way were subcomponents have to announce their intent to process information while a controlling system component tells them if they are allowed to do that or not.

We agree with these design principles. However, the CoSy project developed these principles for cognitive robots in general without having in mind the safety of the human user working with the robot. Therefore, we propose three additional design principles that are necessary to ensure the safety of the human:

– **Robustness.** In a complex system for HRI there are many potential sources for errors; at the input and output modules of the robot as well as at the reasoning modules that control the robot's behaviour. For example, one cannot assume that input recognition modules work correctly all the time; in fact, many times the opposite will be true and recognition errors and missing input can be anticipated. From this, two consequences follow to make sure that the safety in the interaction is increased: the

[1]http://www6.in.tum.de/Main/ResearchJast
[2]http://www6.in.tum.de/Main/ResearchJahir

system must be able to compensate for erroneous or missing input, and when errors occur the system has to be able to identify these errors and to develop strategies to solve them.
- **Fast reaction time.** Robots have to process and react fast to the input they are getting from their environment. This has two obvious reasons: on the one hand, humans will only accept robots if they are reacting fast to what they say or do. Most confusion in the interaction between human and robot arises when the robot is either not reacting at all or if it needs too much time to react to the human's utterances. On the other hand, the robot also needs to quickly detect dangerous situations for the human, for example by the use of sensors or when the human says certain keywords that signal the robot to stop.
- **Context awareness.** When the robot processes the input of its sensor modules, it will inevitably have situations in which the input is ambiguous or where it has not enough input information to compute a complete hypothesis of which action to execute next. In these situations, the robot has to make use of context information, for example it could recognise the human's gestures or look at the objects in its vicinity. Therefore, the robot needs suitable representation formats for its knowledge about the world and defined interfaces to the software modules that provide the information about the robot's environment.

The design principles we are discussing here are of course also important for robotics in general, so why are we regarding these principles as crucial for HRI? Especially, when we are talking about robustness and fast reaction time here, we are thinking of specialised forms of these properties that a robot designed for a safe interaction with a human needs. Usually, robustness in an industrial robot means that the robot is working even if parts of the system are malfunctioning. This can for example be reached by duplicating critical system components. What we mean by robustness is that the robot shows a robust behaviour. It must be aware that in the interaction with a human there might be unclear or unexpected situations, to which the robot has to react in a reasonable way, for example by at least not hurting the human.

Similarly, in general every robot should execute the tasks it was programmed for as fast as possible, but that is not what we mean by "fast reaction time". We argue that in order to increase the safety for the human, the robot has to react to unknown situations fast and also with a reaction that takes into account the partially available context of this situation as fast as possible. This means that the robot has to "understand" the current situation rather than to just react to signal input. Also, only with a reasonable fast reaction time of the robot the human can judge the robot's actions, which also increases the safety for the human, as we already mentioned above.

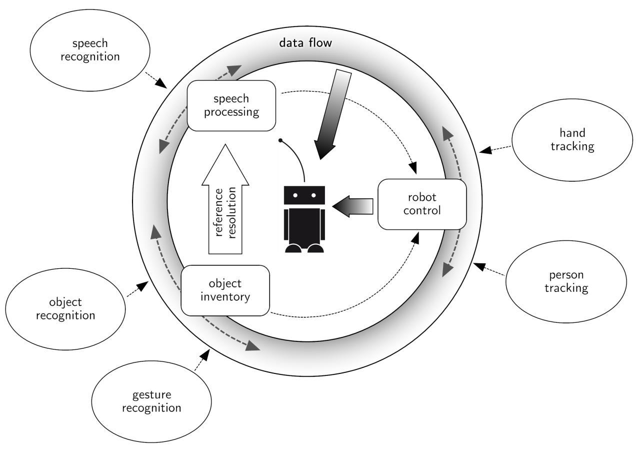

Fig. 1 Robot architecture for safe human-robot interaction.

Therefore, with these design principles we propose a robot architecture that combines approaches that are reaction-based or low-level, which could be seen as part of the embodiment movement as described by Pfeifer and Bongard [33], and high-level methods for reasoning, which are part of more traditional artificial intelligence (AI). This mixture of new and old approaches is also in agreement with Sloman [39], who argues that an architecture for a truly cognitive technical system has to combine methods from embodiment and traditional AI, which also shows similarities to how human perceive and reason about their environment.

Figure 1 shows a schematic overview of our proposed robot architecture. In this architecture, input processing modules, which are depicted by ovals, send the processed data to a central buffer which we call *data flow*. In this schematic architecture overview we show input modules for *speech*, *object*, and *gesture recognition*, and for *hand* and *person tracking*; however, due to the architecture design new input modules can always be added to the system. Reasoning modules, depicted by rounded rectangles, get the data from all input modules that are relevant for them from the data flow. In our example, we have reasoning modules for *speech processing*, which gets data from speech recognition, an *object inventory*, which represents results from object and gesture recognition for reference resolution in the speech processing module, and *robot control*, which uses information mostly from the tracking modules but also gets input from speech processing and uses the object inventory to locate objects in the robot's environment.

After this short introductory overview, we discuss were the design principles for safety in HRI can be found in the architecture and how they affect its design. As the reader will see in the remainder of this publication, we will do this discussion for all the approaches we are presenting here.

Robustness. Robustness seems to be mostly a matter for the input processing modules that have to interpret the robot's environment in a stable way. However, the robot architecture itself can also contribute to increasing the overall robustness of the system. For this, the architecture has to guarantee a robust transmission of information in the data flow as well as to provide mechanisms that help subcomponents of the architecture to recover in case of breakdowns.

Fast reaction time. The robot architecture is important for fast reaction times in two ways: first, without a suitable infrastructure none of the system's subcomponents can provide fast reaction times. The architecture has to take care that information between components flows fast and reliably. Second, the architecture has to provide dedicated fast processing channels for security-related parts of the system. For example, the architecture needs special channels for robot control, which is represented by the two broad arrows in Figure 1. Over these channels the robot can be stopped at all times. This can be done for example from outside the system by pressing an emergency button or by dedicated input processing modules. For instance a specialised module that measures the loudness of the human utterances could stop the robot in case the loudness reaches a certain threshold, which would indicate an emergency situation.

Context awareness. We think that context awareness has to be a built-in feature of a robot architecture in order for the robot to interact with its environment in a reasonable and safe way. Therefore every robot architecture has to be developed already with the context in mind in which the robot has to interact. In our architecture, context awareness is included in two ways: First, the architecture provides the infrastructure for the input modules so that they can publish their recognition results system-wide—and thus in a sense "generate" the context for the robot. For example, the object inventory that represents results from object and gesture recognition is a fixed part of the system. Second, since the safety of the human co-worker of the robot is also part of the context the robot is working in, the architecture has built-in mechanisms to increase the safety for the human. For example, we already mentioned the dedicated subcomponents that are used to stop the robot in case of emergency situations.

Now that we introduced the design principles for safety in HRI and outlined how these apply to robot architectures, we discuss in the following sections how the principles influence our approaches for speech processing (Section 3), vision processing (Section 4), and robot control (Section 5).

3 Speech Processing

Speech is one of the cognitive abilities that sets humans apart from other mammals. Without speech, humans would not be able to joint-actly work together so effectively and safely as they do. Thus, a robot that should interact with a human in a safe way needs to be able to comprehend speech and other human utterances, including gestures or emotions.

However, the problem with natural speech is that humans do not speak grammatically correct and they also tend to leave out parts of the sentences they want to say. Additionally, as Foster et al. showed in [12], many spoken expressions that refer to objects in the world can only be understood together with the gestures that accompany the spoken part of the message. For example, Foster et al. present a real dialogue from a joint construction task, in which two humans assemble tangram models together. The dialogue between the two is shown in Figure 2.

HUMAN 1:	And I'll get this
HUMAN 1:	And then the red one
HUMAN 2:	'Kay I've got the yellow
HUMAN 1:	Cool

Fig. 2 Example of a real dialogue between two humans during a joint construction task.

In the example, the two humans are talking about several triangles, which both of them can see. None of them ever mentions the word triangle, because they can see which object the other person is handling and therefore they know which object the other person is talking about. The dialogue also shows how both of them never say a full grammatically correct sentence.

But how can speech contribute to increase the safety in HRI? If the robot can understand what the human is saying, for example when it knows which object the human is going to pick up next, then it can plan its actions so that it does not necessarily has to operate in the same area as the human. Also, if the robot detects that the human is in the way of its planned path, then it might warn the user by producing an adequate speech output. Additionally, speech is a natural way of communication for humans, so if the robot can understand what the human is saying then this increases the perceived safety feeling of the human.

In speech processing, the main design principle we have to work on is the increase of the robustness of the methods. That starts already with using reliable and fast hardware for speech recognition, but experience shows that the recognition process is always hard to control and that it can produce erroneous input easily. Therefore, the methods for speech processing have to take into account that the input from speech recognition is erroneous and fragmentary. In the next sections, we will explain in more detail our approach for robust speech processing.

3.1 Processing Overview

Our approach for speech processing enables the robot to robustly recognise and understand what the human is saying, to resolve which objects the human is talking about, and to generate an adequate response to the human. In the current version of the approach the robot is not able to have a sophisticated conversation with the human. However, due to the modular setup of the speech processing architecture, the integration of a dialogue manager to the system would be a potential extension in the future.

Figure 3 shows the speech processing subarchitecture for the robot with the single steps of our approach. Rectangular boxes represent processing modules, rounded rectangular boxes are standing for the context information that is available to the processing modules. The single processing steps are as follows: we recognise the speech by a human user with a commercial speech recogniser, in our case Dragon NaturallySpeaking 10. Following speech recognition, the output of the speech recogniser is preprocessed to increase the robustness of the system. After preprocessing, the input is parsed by a grammar to translate it into a logical representation that can be used to resolve the references in the human utterance (i.e. to find out which objects the human is talking about) and to generate a response from the robot to the human. In this response the robot can either confirm that it understood the whole utterance correctly or ask for clarification if it was not able to resolve all references.

The context information that is available to the system consists on the one hand of a grammar that is used in the parsing and output generation steps and from which sentence lists and vocabulary for preprocessing and speech recognition can be automatically generated. On the other hand, the robot gets information from the vision system about objects in the world and about gestures the human made.

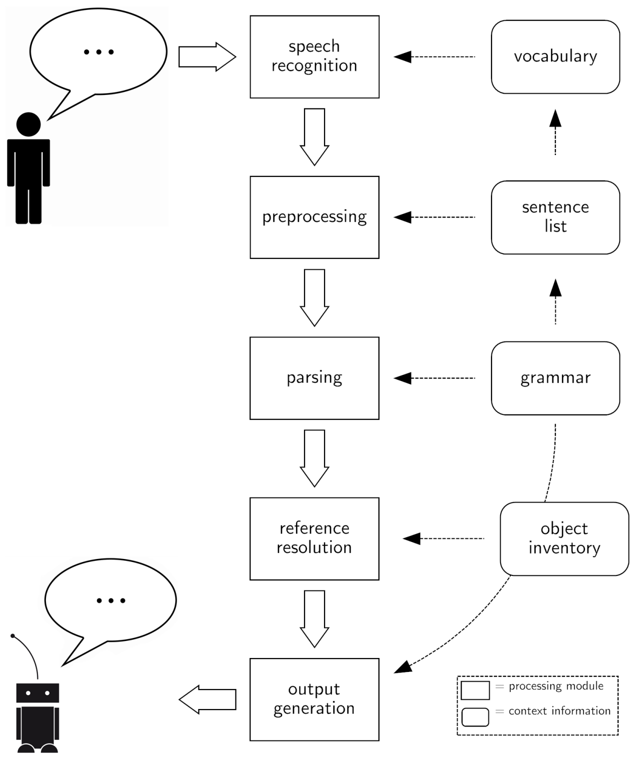

Fig. 3 Speech processing subarchitecture; rectangular boxes stand for processing steps, rounded boxes stand for context information.

In the following sections we will give a more detailed description of the steps for speech processing. Since we use a commercially available software for speech recognition as already mentioned above we will skip speech recognition and start this description with the second step in the processing chain.

3.2 Preprocessing

In our speech processing approach we neither want to limit the words or sentences the human can say to the robot nor want to give the users instructions on what they can say. However, we still want to be able to use a grammar to parse the human utterances, which has many advantages over simpler approaches such as keyword spotting. For this reason, we transform the input by the human into sentences which a grammar can parse. This way, we can use a grammar formalism that has already been proven to work well for human-robot interaction (e.g. in JAST [17], and CoSy [26]) with only slight adjustments to the approach.

For the input transformation, we compare the input sentence we got from speech recognition with a set of

input	“take a”	“take take a cube”
take a cube	1	1
take a bolt	1	2
take a green cube	2	2
take a yellow cube	2	2
take a red cube	2	2
take a green bolt	2	3
take a yellow bolt	2	3
take a red bolt	2	3

Fig. 4 Sentence list with Levenshtein distances for the input sentences “take a” and “take take a cube”.

sentences that can be generated automatically with the grammar we are using in the parsing step (see Section 3.3). For the sentence comparison we currently use an extended Levenshtein algorithm [28]. Figure 4 shows an example for two input sentence comparisons.

The left column of the table shows the list of sentences that has been generated with a grammar. The two columns on the right contain the distances of the two input sentences “take a” and “take take a cube” to the list of sentences. For example, the distance between “take a” and “take a cube” is 1 because we need one insertion to yield “take a cube” from “take a”. The two input sentences show good examples for typical errors in speech recognition, which are missing words or recognising short words twice.

In the example, the two most likely full sentences that would be chosen for input sentence 1 “take a” are “take a cube” and “take a bolt”. In this case, the robot would have to ask the user for clarification which object it should grasp or get the missing information from some other modality. However, since it occurs quite often that some information in the input sentence is missing, we added sentences with placeholders, which we call *empty words*, to the list of sentences. In Figure 5 you can see the list of sentences extended by the sentences with empty words and their distances to the two input sentences from the example in Figure 4.

If we compare the input sentence “take a” to the extended sentence list that contains the empty words we get three sentences that have distance 1 to the input sentence. One of these sentences is “take a emptyNoun” which would be chosen for parsing as we will show in the next section.

3.3 Parsing

The preprocessing step yields an input sentence that can be parsed by a grammar. In our speech processing implementation, we are using combinatory categorial grammar (CCG) to parse and represent speech input. CCG was introduced by Ades [1] and Steedman [40]. It is an extension to the categorial grammar, which is also

input	“take a”	“take take a cube”
take a cube	1	1
take a bolt	1	2
take a green cube	2	2
take a yellow cube	2	2
take a red cube	2	2
take a green bolt	2	3
take a yellow bolt	2	3
take a red bolt	2	3
emptyVerb a cube	2	2
take a emptyNoun	1	2
take a emptyAdjective cube	2	2
emptyVerb a emptyAdjective cube	3	3
emptyVerb a emptyAdjective emptyNoun	3	4

Fig. 5 Extended sentence list with *empty words* and Levenshtein distances for two input sentences.

called lexicalised grammar, of Ajdukiewicz [2] and Bar-Hillel [4]. Traditional context-free grammar formalisms use a top-down approach for parsing sentences, while combinatory grammars utilise a bottom-up approach, which brings advantages in computability and grammar development. Due to the addition of combinatory logic to the grammar formalism, CCGs produce a semantic representation of a sentence during the parsing process. We are using a CCG that was implemented with OpenCCG. OpenCCG [43] is a Java-based implementation of the CCG formalism. It is capable of both, parsing and realising sentences; that means it can translate utterances into a logical form as well as take a given logical form and convert it back to a sentence. OpenCCG generates hybrid logic expressions for the parsed sentence instead of combinatory logic, as explained in [3]. Figure 6 shows such a hybrid logic formula that was parsed with our grammar and represents the sentence “*take this yellow cube.*”. The grammar we are using for our current work is based on a grammar that was developed for the JAST project.

$$
\begin{array}{lll}
@_{t1:action}(\text{take-verb} \wedge & & \\
\quad \langle\textsc{mood}\rangle\ \text{imp} \wedge & & \\
\quad \langle\textsc{Actor}\rangle\ x1 : \mathit{animate-being} \wedge & & \\
\quad \langle\textsc{Patient}\rangle\ (\ c1 : \mathit{thing} \wedge \text{cube-np} \wedge & & \\
\qquad \langle\textsc{det}\rangle\ \text{dem-prox} \wedge & & \\
\qquad \langle\textsc{num}\rangle\ \text{sg} \wedge & & \\
\qquad \langle\textsc{HasProp}\rangle\ (\ y1 : \mathit{proposition} \wedge \text{yellow}))) & &
\end{array}
$$

Fig. 6 Hybrid logic formula that was generated with a combinatory categorial grammar for the sentence “*take this yellow cube.*”. In hybrid logic all actions and entities in the sentence have so-called *nominals*, which can be seen as identifiers.

To understand this logic formula, we have to illustrate the two concepts of *nominals* and *diamond operators* that are part of hybrid logics. *Nominals* can be seen as identifiers that are used to name parts of the logical form. In the present case we used nominals to name the actions expressed in a sentence and the entities that are involved in the action. In the example, the nominal *t1:action* is used to name the take action expressed in the sentence, while the two nominals *x1:animate-being* and *c1:thing* name the actor that should execute the requested action and the cube that should be taken, respectively. The use of nominals to identify actions and entities is very useful for reference resolution, as we will see in the next section, which was one reason to use the CCG formalism for our approach.

In the logical formula we can also see the *diamond operators* ⟨MOOD⟩, ⟨ACTOR⟩, ⟨PATIENT⟩, ⟨DET⟩, and ⟨NUM⟩. These operators represent the syntactic properties of the parsed sentence, including such information as that the sentence was uttered in imperative mood or that a proximal demonstrative was used as determiner to further specify a certain cube. For a more detailed description of the use of CCG, OpenCCG, and hybrid logic, please refer to [17].

Our grammar contains the vocabulary that is necessary for the domain in which our robot is working. This vocabulary consists of verbs that express the robot actions, nouns that describe the entities (i.e. humans, robots, and objects) in the environment of the robot, as well as adjectives that describe properties of the entities (e.g. colours and shapes). As we have already started to explain in the previous section, we added so-called *empty words* to the grammar. These words stand for actions, entities and properties that have not been recognised correctly. For example, the logical form in Figure 7 expresses the fact that the user requested the robot to take an object that is not further specified at the moment. This is signalled by the keyword *emptyNoun-np*.

$$\begin{array}{l}
@_{t1:action}(\text{take-verb} \wedge \\
\quad \langle \textsc{mood} \rangle \ \text{imp} \wedge \\
\quad \langle \textsc{Actor} \rangle \ x1 : animate - being \wedge \\
\quad \langle \textsc{Patient} \rangle \ (\ e1 : thing \wedge \text{emptyNoun-np} \wedge \\
\qquad \langle \textsc{det} \rangle \ \text{indef} \wedge \\
\qquad \langle \textsc{num} \rangle \ \text{sg} \ ))
\end{array}$$

Fig. 7 Logical form with an empty object that has not been recognised correctly.

With the addition of empty words, the robot is able to parse utterances by the human even if the sentence was not recognised correctly. However, the robot then needs strategies to resolve the actions and entities in the logical form as we will see in the next section.

3.4 Reference Resolution

As it was shown in the former section, the parsing step generates a logical form of the spoken utterances that contains the structure of the sentence as well as markers (nominals) for the actions and entities that are contained in the sentence. Therefore, the logical form is perfectly suited to map all the entities the human is talking about to objects and persons in the world. This process is called *reference resolution* or *binding*.

In our approach, reference resolution follows the following steps: (1) since the structure of the sentence is known, we can determine from it the kind of directive sentence that was uttered. According to Quirk [34] (pp. 827) in the English language there are seven ways to utter directive sentences (or commands) that are relevant to our approach. (2) When the sentence structure has been determined, the entities of the sentence can be extracted, this includes in our approach only physical objects with their properties since our robot is only working with one human and has not to know about any other potential co-workers. (3) In the next step, the logical form has to be analysed if it contains any deictic expressions that are cues for the robot that it has to check for pointing gestures by the human as well. Deictic expressions are for example demonstrative determiners like "this" in the expression "take this yellow cube". (4) Now that all objects and gestures that need to be resolved are known, the object inventory, which was introduced in Section 2, can be used to map objects and gestures respectively. For that the object inventory has interfaces over which other modules can make requests if the robot can see objects or gestures with certain properties in its environment.

Finally, reference resolution generates a hypothesis about the spoken utterance that contains the logical form of the utterance together with the referenced objects. Every hypothesis has one of four types:

– **Resolved.** When a hypothesis is resolved then the system was able to find a mapping for each object the human was talking about.
– **Unresolved.** If a hypothesis is unresolved, then the system was not able to find a mapping for each of the objects. In this case, the system has to ask for clarification, as we will discuss in the next Section.
– **Ambiguous.** When the hypothesis is ambiguous, one or more of the found mappings between talked about objects and physical objects is not clear. This means that there are several objects that match the object properties of the logical form. In that case the robot can either choose one of the objects randomly or also ask for clarification.

– **Conflicting.** The system generates a conflicting hypothesis when the human is talking about an object but pointing to another object that has not the same properties as the object he/she is talking about.

3.5 Output Generation

The application of CCGs in HRI has another advantage: as it was already mentioned in Section 3.3, OpenCCG can produce a natural language expression from a given hybrid logic formula, a process that is called *output generation* or *realisation*. Therefore, the logic formula that was produced in the parsing and reference resolution steps can be transformed to generate an output expression by the robot to the human. In the easiest case, when everything was understood correctly, this can be used to confirm that the order by the human was understood correctly. Since the robot knows the objects it is talking about after reference resolution, it can also refer to these objects by using gestures, for example it could point to the objects or look at them if it is equipped with an animatronic head. This behaviour has positive effects on the acceptance of the robot by the human, which was reported in [13].

Output generation can also be used to ask for specific parts of an utterance that have not been understood correctly during the first three processing steps. At this point, the empty words we introduced in the grammar designate the not understood parts of the logical form and can be filtered out easily. This *clarification* process was recently discussed by Kruijff et al. [25], who specify four forms of clarification requests: *attention* when the communication channel between the two conversation agents has to be clarified, *identification* when the utterance was not understood on an acoustic level, *recognition* which refers to clarification in the analysis of the utterance, and *consideration* that is used to analyse the meaning of an utterance.

With our approach we can solve clarification requests from the *recognition* level, which means that the robot can ask for clarification when it has a lexical problem ("what is a slat?"), a grammatical problem ("should I move the cube itself or move myself to its position?"), or referential problems ("which object should I take?", "what should I do with the green cube?").

3.6 Discussion

After this overview of our approach for speech processing in HRI we will discuss the qualification of this approach to increase the safety for the human. We will show how the safety design principles for robot architectures we stated in Section 2 are applied for speech processing. Additionally, we show how our approach could be improved to increase the safety level for the human even further.

Robustness. In speech processing for HRI, robustness is weak because most of the time methods from computational linguistics cannot be applied, since unlike written language, natural language is often ungrammatical and incomplete. Also, speech recognition is still unreliable and needs a lot of tuning although there have been continuous improvements in the field. In our approach, we react to these facts with the preprocessing step before we parse the input by the human. This allows us to use methods from computational linguistics afterwards (parsing with CCG) that have proven to work robustly and fast. Additionally, we added the empty words to the grammar to cope with the uncertainty that comes with using input from speech recognition. At the moment, we react to the underspecification that is expressed by the empty words with clarification requests by the robot. This ensures that the robot does not execute actions the user did not asked it to do and so increases the safety of the human. However, an improvement to this approach would be the addition of a learning component to give the robot the opportunity to learn new words from the human.

Currently we are working on a more sophisticated method to compare the input sentence to the sentences that can be parsed by the grammar. This method should also consider the semantics of the sentences as well as their validity in the current context. The approach that was introduced by Li et al. [29], which uses semantic nets and corpus statistics for comparison of short sentences, could be a basis for this.

Fast reaction time. The usage of CCG in our approach ensures that the input is processable at all times and processing times are much faster in comparison to other grammar formalisms. However, speech processing in general is far away from reaching real time processing, which is mostly due to the fact that most methods for speech recognition and parsing are based on the assumption that their input are whole sentences. To improve on this problem, incremental speech processing how it is proposed by [26] and [5] could be an answer.

Context awareness. The usage of grammars for speech processing and output generation implicitly provides context information for the robot by defining the vocabulary of the application. Of course, one can argue that this way the vocabulary of the robot is constrained and some information of the original sentence the human said might get lost because it contains words that are not in the sentence list. However, we think that a robot which is programmed for a specific task should also be equipped with a vocabulary for this task. For example a robot that builds cars does not need a vocabulary for hotel reservations. A possible improvement to increase the robot's vocabulary could be the use of WordNet [9], which is a lexical database that stores English words that are stored into sets of synonyms. WordNet could be used to automatically retrieve synonyms for the words in the grammar to increase the robot's vocabulary.

In this section, we showed how the design principles for safety in HRI influences speech processing; in the next section, we switch to another channel of input processing and show how we apply the design principles to vision processing.

4 Vision Processing

Every robot that should safely interact with a human needs to be able to interpret the natural environment of the human in a fast and reliable way. Therefore, we present a multithreaded vision subarchitecture that shows a non-blocking and wait-free behaviour. With this architecture we reach an almost linear speed up in performance, which enables the robot to recognise objects as well as human gestures in realtime.

Specifically, we postulate the following requirements that have to be met by the vision subarchitecture:

– **Object and gesture recognition.** The vision system needs to be able to recognise objects, object assemblies, gestures of the human collaborator (e.g. pointing gestures), and parts of the robot.
– **Parallel processing.** The system has to be capable of exploiting a multithreaded environment, independent from actual hardware based degree of parallelisation.
– **Subarchitecture communication.** The communication of the single parts of the vision subarchitecture needs to be efficient and fast.
– **Publication of recognition results at different processing stages.** Preliminary recognition results from early processing stages need to be published in realtime; final recognition results have to be published as fast as possible, but not necessarily in realtime

The design of our vision subarchitecture is independent from hardware assumptions or operating system specifications. In order to achieve optimal performance, we have to take the requirements of multicore computers into account.

In the following sections, we will first give an overview of the vision subarchitecture in Section 4.1. After that, we explain the single steps of our vision processing approach, which are

– **Examine vision processing steps and data.** We determine a functional decomposition of the overall recognition process. Options for hierarchical organisation of processing modules are discussed as well. In addition, we model occurring data in the identified stages as items or partitions in order to allow simultaneous analysis. This step is discussed in detail in Section 4.2 and Section 4.3.
– **Organise parallel processing.** In this step we decide how to apply the asynchronous concept of communication, how to implement a suitable data structure, and we design a task scheduling algorithm. Section 4.4 provides more detailed information on this.
– **Data maintenance.** Finally, we show a data management layer for storage and maintenance of data. This layer has to provide threadsafe access to data items, therefore necessary operations for processing modules must be designed. Here, synchronisation and consistency issues as well as error management must be taken into account, which is explained in detail in Section 4.5.

To conclude the section about vision processing, we discuss in Section 4.6 how the design principles for safety in HRI influenced our vision processing approach.

4.1 Vision Subarchitecture Overview

In order to reach the goal of realtime visual processing we need to speed up the inter-module communication and simultaneous data access in our vision subarchitecture. We propose a multithreaded vision system based on a high level abstraction from hardware, operating system, and low level vision tasks including morphological operations. This allows us to minimise the overhead for communication tasks, as the amount of data needed to be transferred decreases in an abstract representation with a magnitude of a few hundred.

Figure 8 shows an overview of the computer vision (CV) subarchitecture. On the left, the *processing layer* including its decomposition in the functional domain (see Section 4.2) and data domain (see Section 4.3) is depicted. The right column in the figure shows the *data management layer* (details follow in Section 4.5), which handles internal information representation and flow of the CV subsystem according to the second design principle of CoSy ("structured management of knowledge").

The vision system applies an *asynchronous communication mechanism* (ACM), which leads to a non-blocking behaviour and guarantees the required fre-

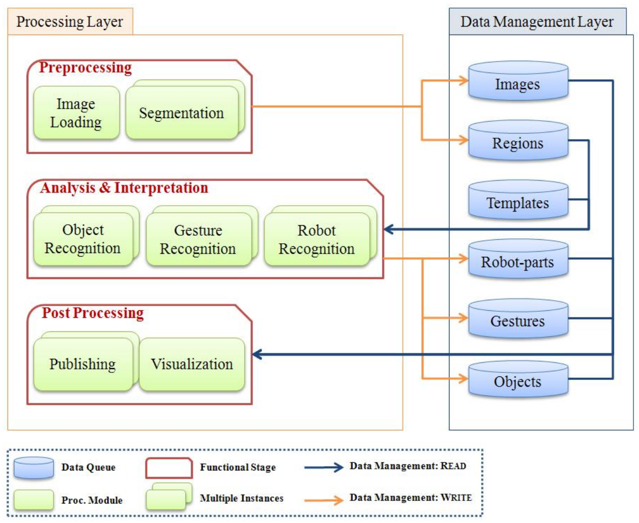

Fig. 8 Vision processing subarchitecture.

quency for result publishing, as publishing incomplete, preliminary analysis results is tolerated (details in Section 4.4). Derived from common standards [11], intermediate vision data is managed in limited-size priority-queues.

4.2 Processing Modules

According to the first design principle of the CoSy project ("concurrent modular processing"), the vision subarchitecture is organised as a combination of functional entities, where each module can operate independently of the others. In this parallel subarchitecture, we have to designate anchor points for distributed computation—the processing modules. Here, the level of abstraction considering computational tasks matters in terms of parallelisation. In order to avoid unnecessary overhead regarding communication and take full advantage of the multicore environment, we model concurrent computation with a moderate level of abstraction.

For function domain parallelisation, we claim that the division into well-defined functional submodules is feasible. In the processing layer of the proposed CV system this is obviously the case, as we can identify three major functional stages: preprocessing, analysis and interpretation, and postprocessing.

Table 1 shows the modulewise decomposition of the CV subarchitecture and lists the data items that are used in the decomposed tasks. As the recognition process is decomposable in the function domain (see **Task** column), now, in order to prove our claim, we must achieve data domain parallelisation. Thus, we have to specify the functional tasks according to the need of multiple instantiation of the processing modules.

Table 1 Functional decomposition of tasks for vision processing.

Task	**Input**	**Output**
Preprocessing		
Image Loading	–	Image
Segmentation / Early Processing	Image	Regions of Attention
Analysis and Interpretation		
Object Recognition	Region and templates	Objects
Gesture Recognition	Region	Gesture
Robot Detection	Region	Robotpart
Postprocessing		
Result Publishing	Objects, gestures and robotparts	–
Result Visualisation	Image, regions, objects, gestures and robotparts	–

4.3 Multiple Instantiation of Processing Modules

Consider that the data management layer successfully stored extracted regions of attention that were extracted by the early processing modules. Analysis operations from the second stage are designed to only alter one specific region (or data item) at a time. Therefore, we can simultaneously serve requests for a region from several concurrently active analysis module instances. Also, the system does not need to wait for the result of an analysis because results are published as soon as they are provided by the data management. We publish periodically in realtime, thus it is of minor relevance to publish a single result at the exact moment of its completed analysis.

Generally, we derive the following approach from the non-blocking paradigm of ACMs: we publish incomplete analysis results of a scene rather than waiting for a complete analysis that would block the system in the meantime. This approach allows us to run multiple concurrent module instances for the analysis of data items as long as the data management is implemented in a threadsafe way, which will be explained in more detail in Section 4.5. Thus, we are able to implement data domain parallelisation in a similar way as it was reported in [6].

4.4 Asynchronous Communication and Scheduling

Originating from functional and data domain decomposition, we have to consider communication strategies for a system running processing modules in parallel. In parallel computation we can generally apply either synchronous or asynchronous communication strategies for data exchange between processes or threads [7]. With synchronous communication the partners wait for confirmation of sent data items, so this strategy has its main applications where the correct transmission of data is essential. Though being robust, due to its blocking nature a synchronous approach can cause problems especially for realtime systems where immediate responses have to be guaranteed.

For this reason, asynchronous non-blocking communication mechanisms (ACMs) have been proposed. With ACMs information or data is dropped when capacities exceed—which is acceptable as long as the system does not block. Non-blocking algorithms can be distinguished as being *lock-free* and *wait-free* [41]. Lock-free implementations guarantee at least one process to continue at any time—though starvation is a risk, because an operation may never finish due to the progress of other operations. On the other hand, wait-free implementations exist that avoid starvation as they guarantee completion of a task in a limited number of steps [23]. Generally, it is essential for systems utilising an ACM to stay responsive, not to guarantee data transmission.

According to Simpson [38], ACMs can be classified based on the destructiveness of data accesses. Table 2 shows protocols with names given by Simpson for different levels of destruction (**N-DR** stands for "non-destructive reading", etc.) and was taken from [37]. Another classification of ACM protocols by Yakovlev [44] distinguishes data access with respect to their overwriting and re-reading permission.

Table 2 Common classification for asynchronous communication mechanisms (ACM).

	DR	**N-DR**
DW	Signal	Pool
N-DW	Channel	Constant

Concerning ACMs for the proposed CV system the implementation of a *pool ACM* in either classification scheme is relevant. On the one hand, we take into account the Simpson classification as we do have non-destructive read operations but write operations include deletion of items. On the other hand, we use Yakovlev's classification, as we allow overwriting in a write operation and do not delete items when reading them from the storage.

There is one disadvantage in such an implementation of the ACM: we risk that a module requests data from the data management, but there is no such data item at the moment, for example because the specific queue is empty. In this case the data management delivers a NULL data item, so modules have to deal with these items as well. Therefore, we propose an algorithm that whenever a NULL data item is received tries to suspend module instances (i.e. single threads) for an optimal amount of time until a correct data item is expected to be delivered again. An incremental back off time $b(c)$ may be calculated for the instance of a module as follows:

$$b(c) = \min(c \cdot i, (\frac{a \cdot j}{n})) \qquad (1)$$

In (1) the parameter c denotes the counter for the number of tries since the last correct data item has been received by the module, i denotes the predefined back off increment in milliseconds, a is the maximum age of a data item until it is deleted, j the number of module instances operating on the same task and n the current number of items in the queue. If a NULL data item is retrieved, c is incremented and the module is immediately suspended for a time $b(c)$ again. In case a correct item could be delivered, c is reset to 0 and the item is processed.

The back off strategy tries to optimally calculate suspension periods for instances not needed at the moment but at the same time to provide an instance whenever needed. In detail, the first argument of *min* calculates an incremental amount of time for the module instance to sleep and the second argument represents the expected mean time until the next correct data item can be delivered. This value is then used as the maximum amount of time to suspend a module instance.

The optimal back off strategy ensures that suspending a single module never blocks the entire vision system, as (1) calculating a maximum for the suspension time guarantees wait-free behaviour (it states an upper limit); and (2) the vision system guarantees, that new camera images are continuously acquired, and thus new data is constantly fed into the system for processing.

4.5 Data Management

A crucial point is the implementation of an adequate data access strategy for concurrent requests. The strategy has to ensure integrity and consistency of data and needs to provide error management policies as well. One also has to consider priorisation whenever a module requests to write while another module simultaneously wants to read data from or write data to the storage. Another important point is the deletion of data items when they expire.

Considering modularity, we organise data access in a data management layer. A natural approach for the implementation is based on the *singleton* design pattern [16]. Singleton implementations only provide a single instance of an object to the overall system; so in our case any request from an analysis module must call the single instance of the data management.

Error handling of the data management layer can be implemented straight forward as the layer delivers NULL data items whenever an erroneous request was received, a queue was empty or no suitable data item could be found. The error handling approach utilising NULL data items is even wait-free because it completes in a limited number of steps [41]. Organising the singleton instance in a threadsafe manner concerning read and write accesses ensures integrity and consistency. In order to achieve this, the data management module is organised as a bundle of queues.

4.5.1 Data Operations

Threadsafe concurrent data access is realised by encapsulating synchronisation. An instance of a processing module sends a request for storage or retrieval of a data item of a certain kind by calling one of the three operations provided to the processing layer. In detail, these operations are defined as follows:

– A WRITE operation requests storage of the passed data item from the data management layer. In pseudo code we can define this access operation as follows:

```
write<Queue>(Item):void
```

– The READ operation requests delivery of a data item from the data management. A READ request enables the module from the processing layer to specify the type of data to be delivered but the concrete item is stochastically determined. The pseudo code notation for this operation is:

```
read<Queue>():Item
```

– The READALL operation requests delivery of all data items from a specific queue. This can be specified as:

```
readAll<Queue>():Item[]
```

4.5.2 Randomised Weighted Data Retrieval

Considering a request from the processing layer, according to what strategy should the data management select an item from the specified queue? The retrieval strategy implemented in data management selects a data item to deliver according to the evaluation of a stochastical function. The function is based on the assumption that a data item (re-)detected in the near past must be prioritised to one that last occurred many cycles ago as it may have already disappeared or removed. Since each item in a queue Q has a timestamp, we weigh the items $i \in Q$ according to their age $a_i = now - timestamp(i)$ such that the weight increases, the younger items are:

$$\forall i \in Q : w_i = 1 - \frac{a_i}{maxage} \tag{2}$$

A new queue of pointers to data items from the original queue is built afterwards. The new queue on which the actual retrieval operation is performed with a random selection is filled with at least one pointer to each data item. In fact, according to the weight w_i of an item i, a number of duplicates d_i of each pointer is pushed to the queue:

$$\forall i \in Q : d_i = \frac{1}{\operatorname{argmin}_{j \in Q}(w_j)} \cdot w_i \tag{3}$$

Afterwards the random selection on the pointer queue is performed where more recent items are prioritised automatically as more pointers to the corresponding data items exist.

Before applying the weight to the items of a queue, we have to exclude elements that match certain preconditions. As an item cannot be altered by two processing modules concurrently, we introduce a *locking* mechanism for items. The non-blocking nature of data access can though still be guaranteed due to the error handling approach described earlier. Before a data item is delivered to the processing layer, the state of the item is changed to *locked*. Locked items are not allowed to be delivered to any other instance and so are excluded from the weighting step. Releasing the lock is in responsibility of the module processing the item.

4.5.3 Queue Locking

Another important question to discuss is the behaviour of the system in case of concurrent WRITE or READ operations concerning a specific queue. Concurrent READ operations are allowed at any time, but in case a WRITE operation is requested all retrieval requests and concurrent WRITE requests must be blocked meanwhile. Therefore, the system has to implement a mechanism that uses cascaded mutual exclusions. Thus, each data queue storing a specific kind of data item is protected by a READWRITE mutex, which exactly matches the above requirements.

Though a single operation may be blocked, the overall system is not. If a mutex can not be acquired at the moment, in case of a READ operation a null item is delivered and in case of a WRITE operation no operation is executed. This behaviour is again conform to the definition of an asynchronous non-blocking algorithm as it is wait-free.

4.5.4 Storage Cleanup

Due to system limitations in a realtime environment we only allow the data management to store a configurable amount of data from the past. Thus, data items must be marked with a timestamp to allow age determination. Consequently there is no DELETE operation provided to the *processing layer*, but items are deleted automatically during a CLEANUP step when they expire. The CLEANUP operation is invoked periodically by a separate data management thread. The thread is scheduled to wait for a certain predefined period of time before it invokes the operation. After performing the operation on all data, the thread is suspended again. Of course, as this operation is equivalent to a WRITE operation, it must be protected by the same mutex mechanism. Similar to the WRITE operation, no operation is performed in case the mutex for a certain queue can not be acquired.

4.5.5 Enhancements

In order to enhance the performance of READ and READALL operations we introduce the concept of *interpretation preselection*. We assume that certain data items are not relevant for dedicated tasks. For example a gesture recognition module could only be interested in a region that comes into the scene from the bottom or an visualisation module is only displaying objects from within the last 100ms, but skipping gestures totally.

In order to completely leave the relevance decision to the processing modules, we propose a mechanism that evaluates a predicate that is passed within a request. According to the predicate, the exclusion step before weighting the items of a queue is adapted. Now not only locked items but also items that do not match the predicate are removed from the set of items from which pointers are duplicated according to their weight. In this way, the search space for retrieval can be restricted, but the non-deterministic selection algorithm can still be applied. Having defined the retrieval method stubs in Section 4.5.1, we now extend these definitions:

```
read<Queue>(Predicate):Item

readAll<Queue>(Predicate):Item[]
```

`Predicate` is a non-empty binary predicate that evaluates to `True` or `False` on each data item of the specified queue. If the set of items after preselection is empty, a NULL data item is delivered. As some items also support state attributes for tracking (*tracked* / *new*) or interpretation (*static* / *moving*), a processing module can use these attributes for the implementation of its own predicate.

Moreover, a processing module can in principle specify arbitrary predicates on members of the data items in a queue. For example, considering that data items in a queue are timestamped, it is possible to *track* them from one cycle to the following. Therefore, we define a comparison predicate that is applied when a storage WRITE query is requested. The predicate evaluates symbolic or meta attributes, including classification, colour, approximate pose, number of points, or width, height, and depth. When the data management receives a WRITE request for a formerly recognised item, only necessary attributes may be updated, all other attributes (especially the unique id) instead can be kept. For example, considering an item static and fully analysed, the existing item just gets all ordinary attributes (such as the timestamp, position, etc.) updated, but the updated item is not marked for analysis again.

4.6 Discussion

Following this overview of our approach for vision processing, we discuss how the design principles for safety in HRI influence the method.

Robustness. To increase the robustness of object and gesture recognition, we are using multiple modalities in the recognition process: the use of colour, shape, and size information makes sure that a good recognition result can be guaranteed at all times, even in cases were one of the object properties cannot be determined correctly.

Fast reaction time. For a decreased processing time we took two actions: first, we make use of the multicore architectures of current computers to parallelise our approach for object and gesture recognition. Because parallelisation scales very well considering the proposed approach, with its introduction, the processing times for analysis of a complete frame could approximately be cut into $1/n$, where $n = 4$ on the quad-core computer utilised [31].

Second, we are using tracking to follow already recognised objects and gestures. If the size and position of an object only changes slightly, we assume that it still is the same object and thus do not initiate the whole recognition process for this object. Considering this, we are able to increase processing times again, as the actual object recognition is the most time-consuming task in the vision system.

As a future step, we plan to realise a mechanism called *attention-based early processing*, which has been introduced in [30]. In this method, object recognition publishes its preliminary recognition results at an early processing stage. This way, the robot already gets information about the rough position and size of an object but does not know the exact type and properties of the object yet. This can for example be used to avoid collision with moving objects.

Context awareness. For object recognition, context is mostly important to further increase the robustness and processing time of the approach. In our case we are using the information about the position of the robot's arm to determine whether it is moving within the field of view of one of the used cameras. In that case, the vision system may interrupt the recognition process which decreases squandering of computing power and avoids false recognition results.

Now that we have highlighted the input side of HRI and showed our work on speech and vision processing, we want to switch to the output side and explain how the design principles for safety influence our approach for robot control.

5 Robot Control

The consideration of the safety design principles is of utmost importance controlling a collaborative robot. Especially working in a shared workspace might sooner or later lead to situations where human and robot collide with each other. That this can be very harmful to the human especially when standard industrial robots are used, was shown in [19,20,18].

In this section, we discuss how the safety design principles influence the controller module of a robot. We are mainly concerned to speed up the reaction times of the robot and to increase the context awareness of the system. For that, on the one hand we are splitting up the robot actions in several tasks that can be stacked up in a task hierarchy. This speeds up the online calculation of the robot movements and allows the robot to be reconfigured for various applications easily. On the other hand, we equip the robot with an internal representation of its environment in which the sensor data from vision processing, person tracking, and hand tracking is updated continuously in an asynchronous fashion, so that the robot always knows about its surrounding.

In the remainder of this section we show in Section 5.1 how single robot tasks, including posture, operational position, and collision avoidance, can be arranged in a task hierarchy so that the safety for the human is increased without constraining the robot's capabilities. Section 5.2 gives an overview of the internal robot representation, and Section 5.3 presents how our approach for robot control can be applied in an application for HRI in an industrial setting. Section 5.4 concludes the section about robot control and discusses how the safety design principles influence our approach.

5.1 Robot Task Hierarchy

According to [36], actions consist of several atomic tasks that are arranged in a task-oriented way. That means on the reverse that various different actions can be generated by rearranging the same set of atomic tasks. However, an encoding of priorities in the task hierarchy is of importance, because it needs to be prevented that contrary tasks interfere with each other and lead to uncontrollable or unwanted behaviour of the robot, which might lead to situations in which the safety of a human is in danger.

Based on the syntax of [45], an action A can be formulated as a composition of tasks T_k with a projection rule $\lhd_k$ that ensures the behaviour of the task:

$$A = \langle T \rangle_0^n = T_n \lhd_n T_{n-1} \lhd_{n-1} \dots \lhd_1 T_0. \qquad (4)$$

As most industrial robots are only controllable on the position or velocity level, we transform the task descriptions in terms of joint velocities ($\underline{\dot{q}}$):

$$\underline{\dot{q}}_{\text{Action}} = \underline{\dot{q}}_{T_n} \lhd_n \underline{\dot{q}}_{T_{n-1}} \lhd_{n-1} \dots \lhd_1 \underline{\dot{q}}_{T_0}. \qquad (5)$$

According to the defined control structure, we need to take care of only a few points for each task: compute the velocity to solve the single task, set the constraints according to the current situation or use static constraints, and finally transform the lower priority task velocity into a safe subspace respecting the active constraints. For the latter we are using *null spaces* with orthogonal projectors as in [36,35] in the posture and the operational position task and a constraint least square optimisation for the collision avoidance task. With these projectors we are able to decouple the tasks from each other which is also shown in Figure 9.

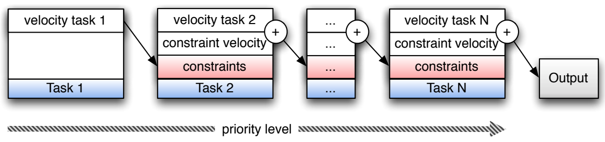

Fig. 9 Actions are defined through task compositions. The associated constraints are respected through projections into corresponding subspaces, with execution priorities ranging from lowest (left) to highest (right).

5.1.1 Respecting Hardware Limits

An important issue that needs to be respected are the limitations of joint angles, velocities, and accelerations. The resulting velocity $\underline{\dot{q}}$—as well as intermediate results—needs to be in a certain bounding region $\underline{l} \leq \underline{\dot{q}} \leq \underline{u}$, that does not violate these limits. Therefore, we adaptively recompute the bounding limits in every time step according to

$$\underline{l}_i = \max(\dot{q}_i^{\min}, \dot{q}_i^{\text{lowerJL}}, \dot{q}_i^{\text{AL}}) \tag{6}$$

for the lower boundary and

$$\underline{u}_i = \min(\dot{q}_i^{\max}, \dot{q}_i^{\text{upperJL}}, \dot{q}_i^{\text{DL}}) \tag{7}$$

for the upper boundary. The values for $\dot{q}_i^{\max}$ and $\dot{q}_i^{\min}$ are the maximum and minimum velocity for joint i as defined by the manufacturer of the robot.

$$\dot{q}_i^{\text{lowerJL}} = \frac{q_i^{\min} - q_i(t)}{\Delta t} \tag{8}$$

is the velocity that is needed to reach the lower joint limit of joint i in the next time step $t+1$ and

$$\dot{q}_i^{\text{upperJL}} = \frac{q_i^{\max} - q_i(t)}{\Delta t} \tag{9}$$

is the velocity to reach the upper joint limits respectively. These velocities converge to zero as the joint angle is approaching the joint limit.

To respect that the acceleration and the deceleration abilities of the motors of the robot are also limited, the maximum velocities of the joints need to be constraint. Therefore, we approximate this factor with

$$\dot{q}_i^{\text{AL}} = \underline{\ddot{q}}_i^{\text{acc}} \cdot \Delta t + \dot{q}_i(t-1) \tag{10}$$

for the acceleration and

$$\dot{q}_i^{\text{DL}} = \underline{\ddot{q}}_i^{\text{dec}} \cdot \Delta t + \dot{q}_i(t-1) \tag{11}$$

for the deceleration of the joints.

With the limit information for each joint, a limit method $C(\underline{\dot{q}})$ can be defined that scales the velocity vector to respect all above mentioned limits *without* changing the trajectory of the motion:

$$C(\underline{\dot{q}}) = s(\underline{l}, \underline{u}) \cdot \underline{\dot{q}}. \tag{12}$$

In the following sections, we explain the tasks *posture, operational position*, and *collision avoidance*. These tasks can be stacked in an arbitrary hierarchy to perform various actions.

5.1.2 Task: Joint Position (Posture)

The goal of the posture task is to set the robot to a certain joint configuration $\underline{q}_{goal}$. With this posture controller the robot can be set to specific postures, for example to a “human friendly” or upright posture.

The velocity for this task is calculated by

$$\underline{\dot{q}}_{po} = C\left(\frac{\underline{q}_{goal} - \underline{q}(t)}{\Delta t}\right). \tag{13}$$

To constrain certain degrees of freedom in joint space that cannot be influenced by lower priority tasks a $n \times n$ matrix S_{po} is used to select them, with n being the number of joints. This means the guaranteed velocity can be expressed as

$$\underline{\dot{q}}^*_{po} = S \cdot \underline{\dot{q}}_{po}. \tag{14}$$

Using the projector N_{Po} of the selected constraints, we get as output for the projection of an input velocity $\underline{\dot{q}}_{in}$:

$$\underline{\dot{q}}^* = \underline{\dot{q}}^*_{po} + N_{po} \cdot \underline{\dot{q}}_{in} \tag{15}$$

with

$$N_{po} = I - S_{po}. \tag{16}$$

5.1.3 Task: Operational Position

The operational position task drives the tool centre point of the robot to a defined goal position and orientation $\underline{x}_{goal}$ in Cartesian coordinates according to:

$$\underline{\dot{x}} = \frac{\underline{x}_{goal} - \underline{x}(t)}{\Delta t}. \tag{17}$$

After calculating the velocity in Cartesian space, constraints can be set using a diagonal selection 6×6 matrix S_{Op} to select the degrees of freedom (in Cartesian space) that should not be influenced by lower priority tasks. Transformed into limited joint velocities using a singularity robust pseudo-inverse $J_e^\dagger$ of the Jacobian J_e of the end effector, we get

$$\underline{\dot{q}}^*_{op} = C(J_e^\dagger \cdot S_{op} \cdot \underline{\dot{x}}_{op}). \tag{18}$$

Using the null space N_{op} of the selected constraints, we get as output for the orthogonal projection of an input velocity $\underline{\dot{q}}_{in}$:

$$\underline{\dot{q}}^* = \underline{\dot{q}}^*_{op} + N_{op} \cdot \underline{\dot{q}}_{in} \tag{19}$$

with

$$N_{op} = I - J_e^\dagger S_{op} J_e. \tag{20}$$

5.1.4 Task: Collision Avoidance

With this task, the robot can be controlled so that it avoids collision with static entities (for example a workbench) and dynamic entities (for example the human or moving objects) in its environment. In this task, the avoidance is done in a reactive way with dynamically updated collision scenes that is interfaceable with a variety of sensors. The vision system we presented in Section 4 also connects to the robot controller via this interface.

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

Virtual Forces

To compute the velocity that repels the robot from surrounding obstacles, we need to compute the minimum distances of all objects in the environment model (including self collision) to all body parts of the robot. Figure 10 shows the body parts of the robot we are using in our examples in different colours along with an example of the minimum distances d_i (red lines) from an obstacle to a given joint configuration.

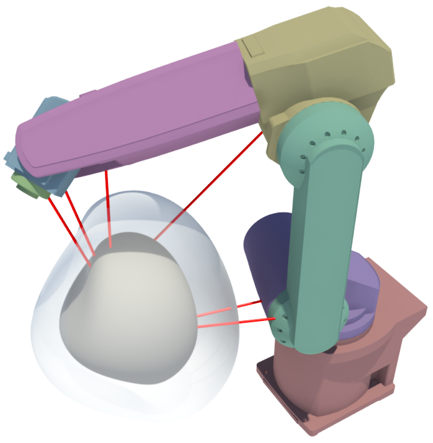

Fig. 10 Computing the repelling forces of an obstacle: the red lines illustrate the minimum distances of an obstacle to the body parts of the robot in a given joint configuration. If a distance is below a chosen security threshold (transparent bubble), the distance is used to compute virtual forces on the robot using potential fields.

Opposite to simplified and only approximated models of manipulators (which was for example used in the skeleton algorithm presented in [8]), we measure the distances of arbitrary shapes to a convex version of the real CAD model of the robot in order to reach a high precision of the virtual forces. With an efficient implementation of the GJK algorithm [42], we compute these distances 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), we need to transform them to velocities in joint space and find the overall motion of the robot to avoid the collision. This is done according to

$$\underline{\dot{q}} = \sum_{i}^{I} \underline{\dot{q}}_i = \sum_{i}^{I} J^T_{P_r(i)} \cdot U_{rep,i}(q) \cdot v_{x,i}(q), \tag{21}$$

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 $U_{rep,i}$:

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

with Q^* being the distance at which the potential field function is applied, which is represented by the transparent bubble in Figure 10.

Constraint Least Square Minimisation

To ensure that we *project* the lower priority task in an orthogonal subspace of the collision avoidance task, we use the mathematical framework of quadratic programming [10] to minimise 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 optimised regarding *must have* constraints of the higher priority task. The projection is described according to:

$$\min_{\underline{\dot{q}}_{in}} \| J_e \cdot \underline{\dot{q}}_{in} - \underline{\dot{x}}_t \|^2, \tag{22}$$

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

$$C^T \underline{\dot{q}}_t \ge 0 \tag{23}$$

with the constraint matrix

$$\underline{C}^T = \begin{pmatrix} \underline{\dot{q}}_1^T \\ \vdots \\ \underline{\dot{q}}_I^T \end{pmatrix} \tag{24}$$

and $\underline{\dot{q}}_i$ calculated according to (21). This means, only those velocities are valid, which 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 minimisation process is then equal to the constrained joint velocity $\underline{\dot{q}}^*$. Quadratic Programming in combination with collision avoidance on the level of joint acceleration was used in [15,14].

5.2 Internal World Representation

The robot controller has an internal representation of its surrounding environment (internal world representation, IWR) that is kept up to date with changes in the world that are of interest for the robot.

Since the robot in the shown collaboration scenario is stationary, many things in the environment can be added to the IWR as static components. This includes the worktable, the cage, the assembly belt in the background, as well as the robot's position in relation to the table. Multiple sensor modules perceive dynamic objects—including the human—and add, update or remove them at runtime. Object properties such as the colour or the location of the object, as well as the robot's posture, which is also constantly updated when the robot moves are also part of the IWR.

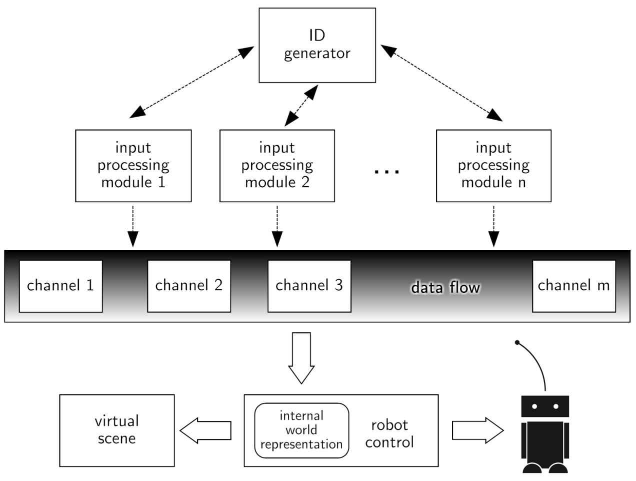
ID generator
input processing module 1
input processing module 2
...
input processing module n
channel 1
channel 2
channel 3
data flow
channel m
virtual scene
internal world representation
robot control

Fig. 11 Schema for the internal world representation component.

Figure 11 shows a schematic overview of this representation. The *ID generator* is a central component of the IWR. *Input processing modules*, for example an object recognition or a person tracking module, request globally unique IDs from the generator. With these IDs the input processors can introduce, update, and remove new entities (i.e. persons or objects) in the IWR. Here, the input processing modules have to organise the IDs they are using by themselves, which also means that they have to take care that they remove entities from the IWR when they do not track them anymore. Figure 5.2 shows the interfaces for handling entities in the IWR.

```
addSingleObject(groupID, bodyID, shape, pose);
updateSingleObject(bodyID, pose);
removeSingleObject(bodyID);
addGroupObject(groupID, object, pose);
updateGroupObject(groupID, pose);
removeGroupObject(groupID);
```

Fig. 12 Interfaces for the internal robot representation; input processing modules use these interfaces to add, remove, or update entities in the representation.

Robot control uses the IWR on the one hand to keep track of the current status of the environment. On the other hand, it also forwards the information from the input processing modules to a virtual scene, in which the controller also presents information about the robot's appearance and position. For this, the virtual scene first loads information about the static environment of the robot and then gets the additional information about dynamic entities in the world from the controller. This virtual scene can be displayed to the human users so that they also get a quick overview of the robot's belief about the current state of the world. Figures 13(b) to 13(d) show an example for a 3D representation of a virtual scene.

5.3 Application Example: Mobile Storage Box

With the task-based hierarchical control structure we can solve a wide range of tasks that can be used in production scenarios in which human and robot work together as a team. In the following section, we present an application of our robot control approach that was used on the demonstration platform JAHIR (Joint-Action for Humans and Industrial Robots) [27], which is embedded in a factory setting [46].

The JAHIR platform consists of an industrial robot arm that is mounted behind a workbench. A human who stands face to face to the robot shares the workspace with the robot during collaborative tasks. Figure 13(a) shows the system interacting with a human. During interaction, the human wears a jacket on which a set of infrared markers are attached on the palm and on the lower and upper arm. An infrared tracking system tracks these markers and sends the sensor data to the internal representation of the robot. This way, the robot constantly updates its knowledge about the position of the human's arm and hand. Figures 13(b) to 13(d) show a 3D model of the internal robot representation in various interaction situations. In the representation, the human's body is roughly estimated by several cylinders which ensures that the human's safety zone is big enough.

Our application example is set in a manual production setting. In manual production human workers need to have all parts they need for an assembly within reach so that they can build it efficiently. At the same time, the workers need a place were they can put already assembled subcomponents of an assembly, from which they can retrieve the subcomponent in a later production stage.

Thus, we programmed the JAHIR robot to assist the human as a mobile storage box. The robot is able to supply assembly parts as well as to take already finished subcomponents of the assembly and to keep them in range of the human. For this, the robot holds a storage box and keeps it near the human's hand. To do this in a safe way, the robot has to bring in line the tasks to follow the human's hand and to avoid collision with the human and with its environment at the same time. Given these requirements, we configure the robot controllers as presented in Equation (25) to compose action A_1.

$$A_1 = T_{\text{orientation}} \lhd T_{\text{avoidance}} \lhd T_{\text{position}} \lhd T_{\text{posture}}. \tag{25}$$

$T_{\text{orientation}}$ is the task with the highest priority, which takes care of keeping the box always in a horizontal orientation. The operational position controller is used here with the selection matrix

$$S_{\text{orientation}} = \text{diag}\,(0, 0, 0, 1, 1, 1) \tag{26}$$

to fix the orientation of the box. Task $T_{\text{avoidance}}$ avoids collisions with the surrounding environment and the human hand. The position task T_{position} follows the human hand through updates of the hand tracking system to the goal position x_{goal}, that should be 0.1 m in front and below of the hand. To keep the position fixed, the selection matrix

$$S_{\text{position}} = \text{diag}\,(1, 1, 1, 0, 0, 0) \tag{27}$$

is used. In the posture task, we defined that the robot should have an upright joint configuration. Because the posture task has the lowest priority, we can include all joints in the velocity calculation.

The orientation has the highest priority in this example, because the spare parts, that are inside the box, should not fall to the ground. Although this seems to limit the safety for the human at first glance, each task in the controller can suspend the whole movement of the robot. This means, that if the human comes to close to the robot and the collision shapes touch, the robot stops until the collision is cleared. The collision shapes have a bigger size than the real objects, for example the hand is approximated as a sphere. Another issue that should be considered is, that although the orientation is fixed with highest priority, the robot has three positional degrees of freedom left to avoid the collision.

Figure 13 shows some impressions from the mobile storage box application. The robot is carrying a red box with assembly subcomponents in its gripper, so that the human can retrieve parts of the box, which is shown in Figure 13(a). In order to allow the human to grasp a subcomponent from the box, the motion of the robot needs to be stopped. For this the robot controller uses information from the avoidance task that measures the distance between robot and human. The controller stops the current motion, if the distance of robot and human falls below a defined threshold.

5.4 Discussion

To conclude the section about robot control, we discuss how the design principles for safety in HRI influence our approach. Additionally, we show how the approach can be even further improved.

Robustness. To increase robustness in robot control, we take three steps: first, we compute the trajectories of the robot actuator online every seven milliseconds. This way, the robot can react quickly on new events sent by the input processing modules in an asynchronous fashion. Second, the decomposition of robot tasks into single actions decreases the complexity of the trajectory computation and therefore increases robustness. Third, in the internal world representation we chose shapes to represent human body parts, which are bigger than the body part in reality. Together with the computed virtual repelling forces for collision avoidance, this ensures that the robot stops its movement long before an actual collision can occur. All tasks can suspend the motion of the robot, for example when a collision or an error in the computation occurs.

To further improve our approach and to increase the stability of the collision avoidance, we plan to use more distances than only the minimum distance to compute the virtual forces. Additionally, we want to add more atomic tasks to the controller, including singularity and joint limit avoidance, and we plan to track the whole human body pose at best without the use of any markers.

Fast reaction time. Fast and efficient computation is very important for robot control. Therefore, in

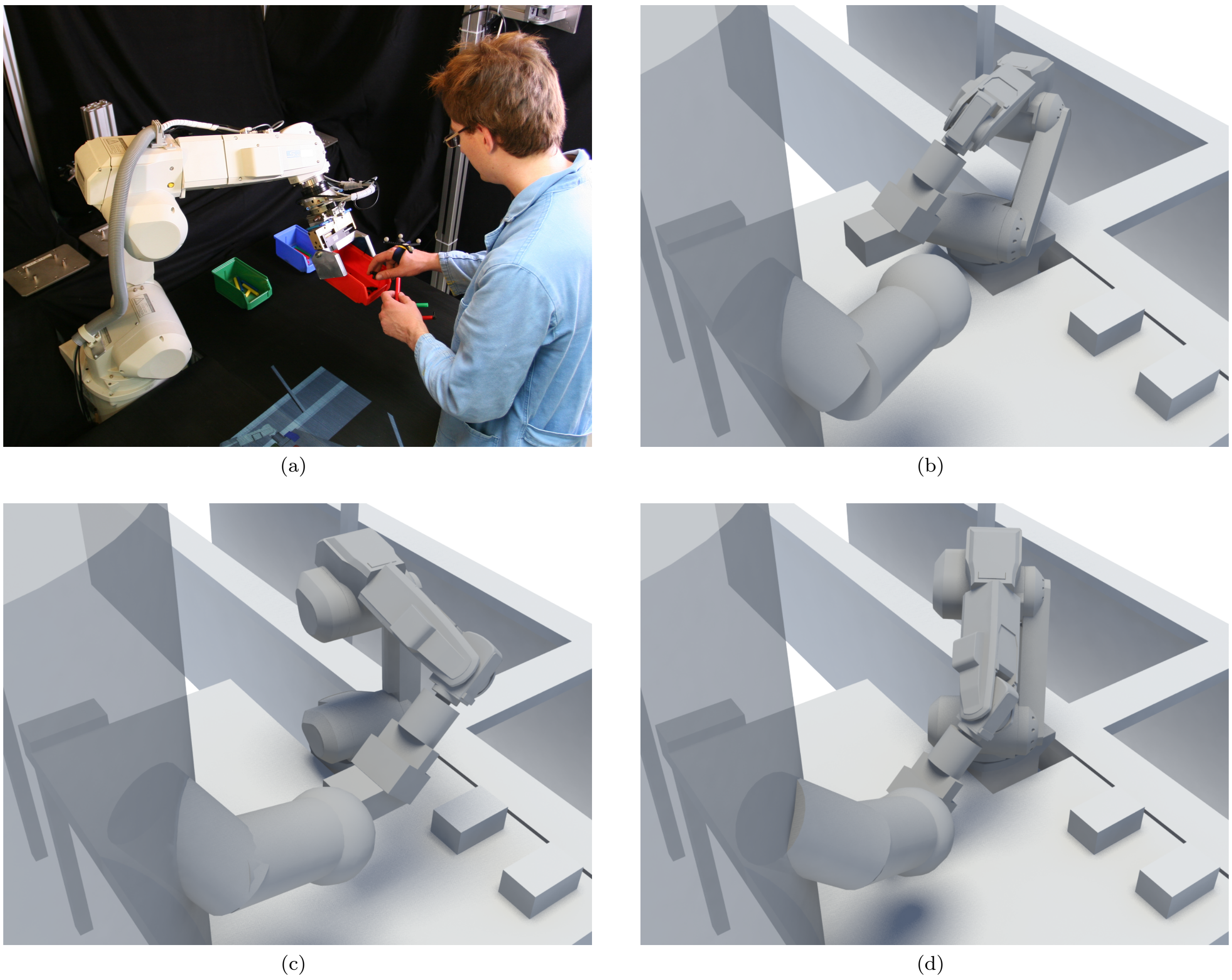

Fig. 13 Mobile storage box: The robot has a storage box as tool which follows the human hand in safe distance and avoids collision with the hand and the surrounding, so that the human can pick up parts or place assembled products in there. (a) real scene; (b), (c) and (d) 3D representation with robot behaviour.

our approach we compute robot trajectories faster than the robot update time, even on a standard dual core personal computer. To improve our approach in the future, we plan to parallelise the computation by sourcing out computationally intensive parts of the approach—e.g. computation of collision avoidance—to other computers.

Context awareness. We increased the context awareness in our approach by adding the internal world representation to robot control. This way, the robot always has an exact information about the current status of its environment and can make sure that it does not collide with a human or any objects.

In the next step, we plan to increase the context awareness of the robot even further by adding full body tracking of the human to the internal world representation. Additionally, we plan to equip the setup with a depth sensor camera. This will enable the robot to measure the size of objects, even if they are unknown, so that it can avoid collision.

6 Conclusion

In this publication, we presented three design principles for safety in human-robot interaction, *robustness*, *fast reaction time*, and *context awareness*, which are essential for every software component that controls the behaviour of a robot that interacts with a human. We showed how these design principles can be applied to robot architectures and how they influence approaches for speech processing, vision processing, and robot control. We also presented an example application that incorporates these design principles. In this application, a human works together with a robot that assists the human as a mobile storage box. Due to the effective implementation of the design principles, the efficiency of the human worker is increased without having to decrease the safety level for the human.

The main contributions of our work, which all increase the safety for the human that interacts with a robot are as follows: we proposed a method for speech processing that enables the robot to deal with situations in which it did not understand the whole utterance of its human collaborator completely, which increases the robustness of speech processing. In vision processing we presented a new algorithm for recognition of objects and gestures, which makes use of modern multicore computers and guarantees a fast, reliable, and non-blocking recognition. Finally, we showed how the robot can be programmed with a hierarchical task structure, which enables the robot to fulfil its assignments while making sure that the safety of the human is guaranteed at all times.

Our design principles are one step towards increased safety in HRI and we showed how we apply these principles to several parts of a robot system. However, in the next step we have to analyse how the design principles influence other components of HRI systems, especially reasoning components such as task planners and dialogue managers. Additionally, we have to find out how the combination of several approaches for safety in HRI can be combined, for example a combination of specialised safe robotics hardware with standard industrial robots would be interesting.

Another aspect of safety in HRI—which we did not touch in this publication at all—is how humans perceive robots and how their perceived feeling of safety can be increased. For this, it has to be studied how the robot's motions (trajectories and speed) influence the human's acceptance rate. For example Huber et al. [24] showed that humans report an increased level of perceived safety when the robot uses human-like motion trajectories. Also, we need to determine what the robot should say or do in order that the human knows the robot's next actions. Overall, we are convinced that the combination of the approaches of other researchers and the approaches we are proposing here will lead to a boost for more human-robot interaction in industrial settings and also in assistive robotics.

Acknowledgements This work is supported by the German Research Council (DFG), under the excellence initiative cluster "Cognition for Technical Systems" (CoTeSys[3]) and the Collaborative Research Center SFB 453 on "High-Fidelity Telepresence and Teleaction".

[3]http://www.cotesys.org

References

1. A. E. Ades and M. J. Steedman. On the order of words. *Linguistics and philosophy*, 4:517–558, 1982.
2. K. Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.
3. J. Baldridge and G.-J. Kruijff. Coupling ccg and hybrid logic dependency semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 02)*, Philadelphia, PA: University of Pennsylania, 2002.
4. Y. Bar-Hillel. A quasi-arithmetic notation for syntactic description. *Language*, 29:47–58, 1953.
5. T. Brick and M. Scheutz. Incremental natural language processing for hri. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 263–270. ACM New York, NY, USA, 2007.
6. T. P. Chen, D. Budnikov, C. J. Hughes, and Y.-K. Chen. Computer vision on multi-core processors: Articulated body tracking. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1862–1865. Intel Corporation, IEEE ICME, 2007.
7. D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach.* Morgan Kaufmann Publishers, 1999.
8. A. De Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, and 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*, pages 1–6, 2007.
9. C. Fellbaum et al. *WordNet: An electronic lexical database.* MIT press Cambridge, MA, 1998.
10. R. Fletcher. Quadratic programming. In *Practical Methods of Optimization*, chapter 10, pages 229–258. Wiley, New York, second edition, 1987.
11. Message Passing Interface Forum. Mpi, a message-passing interface standard. Technical report, University of Tennessee, Knoxville, Tennessee, June 1995.
12. M. E. Foster, E. G. Bard, R. L. Hill, M. Guhe, J. Oberlander, and A. Knoll. The roles of haptic-ostensive referring expressions in cooperative, task-based human-robot dialogue. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction (HRI 2008)*, pages 295–302, Amsterdam, March 2008.
13. M. E. Foster, M. Giuliani, A. Isard, C. Matheson, J. Oberlander, and A. Knoll. Evaluating description and reference strategies in a cooperative human-robot dialogue system. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, California, July 2009.
14. E. Freund and 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*, pages 1173–1177, 2003.
15. E. Freund, M. Schluse, and J. Rossmann. Dynamic collision avoidance for redundant multi-robot systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1201–1206, 2001.
16. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Orientated Software.* Addison-Wesley Professional Computing Series, 1998.

17. M. Giuliani and A. Knoll. Integrating multimodal cues using grammar based models. In Constantine Stephanidis, editor, *Proceedings of the 4th International Conference on Universal Access in Human-Computer Interaction, HCI International, Part II*, volume 4555 of *Lecture Notes in Computer Science*, pages 858–867, Beijing, July 2007. Springer.
18. S. Haddadin, A. Albu-Schäffer, M. Frommberger, J. Rossmann, and G. Hirzinger. The "dlr crash report": Towards a standard crash-testing protocol for robot safety-part ii: Discussions. In *IEEE Int. Conf. on Robotics and Automation (ICRA2008), Kobe, Japan*, pages 280–287, 2009.
19. S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Safety evaluation of physical human-robot interaction via crash-testing. In *Robotics: Science and Systems Conference (RSS2007)*, pages 217–224, 2007.
20. S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. The role of the robot mass and velocity in physical human-robot interaction-part ii: Constrained blunt impacts. In *IEEE Int. Conf. on Robotics and Automation (ICRA2008), Pasadena, USA*, pages 1339–1345, 2008.
21. S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *International Journal of Robotics Research*, 2009.
22. N. Hawes, J. Wyatt, and A. Sloman. An architecture schema for embodied cognitive systems. Technical Report CSR-06-12, University of Birmingham, School of Computer Science, November 2006.
23. M. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, 1991.
24. M. Huber, H. Radrich, C. Wendt, M. Rickert, A. Knoll, T. Brandt, and S. Glasauer. Evaluation of a novel biologically inspired trajectory generator in human-robot interaction. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, September 2009.
25. G.-J. Kruijff, M. Brenner, and N. Hawes. Continual planning for cross-modal situated clarification in human-robot interaction. In *Proceedings of the 17th International Symposium on Robot and Human Interactive Communication (RO-MAN 2008)*, Munich, Germany, August 2008.
26. G.-J. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, and N. Hawes. Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction. In Luis Seabra Lopes, Tony Belpaeme, and Stephen J. Cowley, editors, *Symposium on Language and Robots (LangRo 2007)*, Aveiro, Portugal, December 2007.
27. C. Lenz, S. Nair, M. Rickert, A. Knoll, W. Rösel, J. Gast, and F. Wallhoff. Joint-action for humans and industrial robots for assembly tasks. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*, pages 130–135, 2008.
28. V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710, 1966.
29. Y. Li, D. McLean, Z.A. Bandar, J.D. O'Shea, and K. Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, pages 1138–1150, 2006.
30. T. Müller and A. Knoll. Attention driven visual processing for an interactive dialog robot. In *Proceedings of the 24th ACM Symposium on Applied Computing*, Honolulu, Hawaii, USA, March 2009.
31. T. Müller, P. Ziaie, and A. Knoll. A wait-free realtime system for optimal distribution of vision tasks on multicore architectures. In *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics*, Funchal, Portugal, May 2008.
32. C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schäffer, B. Brunner, H. Hirschmuller, S. Kielhofer, et al. A humanoid two-arm system for dexterous manipulation. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 276–283, 2006.
33. R. Pfeifer, J. Bongard, and S. Grand. *How the body shapes the way we think: a new view of intelligence*. The MIT Press, 2007.
34. R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A comprehensive grammar of the English language*. Oxford Univ Press, New York, 1985.
35. L. Sentis. *Synthesis and Control of Whole-Body Behaviors in Humanoid Systems*. PhD thesis, Standford University, 2007.
36. L. Sentis and O. Khatib. Task-oriented control of humanoid robots through prioritization. In *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, 2004.
37. H.R. Simpson. Methodological and notational conventions in doris real-time networks. In *IED Supporting Predictable Implementation of Requirements in Timing and Safety (SPIRITS) Deliverable*, 1994.
38. H.R. Simpson. Protocols for process interaction. *IEE Proceedings-Computers and Digital Techniques*, 150(3):157–182, 2003.
39. A. Sloman. Some requirements for human-like robots: Why the recent over-emphasis on embodiment has held up progress. In B. Sendhoff, E. Koerner, O. Sporns, H. Ritter, and K. Doya, editors, *Creating Brain-like Intelligence*, pages 248–277. Springer-Verlag, Berlin, 2009.
40. M. Steedman. *The syntactic process*. MIT Press, Cambridge, MA, USA, 2000.
41. H. Sundell and P. Tsigas. Fast and lock-free concurrent priority queues for multi-thread systems. *Journal of Parallel and Distributed Computing*, 65(5):609–627, 2005.
42. G. van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.
43. M. White. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language & Computation*, 4(1):39–75, 2006.
44. A. Yakovlev, F. Xia, and D. Shang. Synthesis and implementation of a signal-type asynchronous data communication mechanism. In *Proceedings of the 7th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, page 127, 2001.
45. Y. Yang and O. Brock. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Proceedings of the Robotics Science and Systems Conference*, 2006.
46. M. Zäh, M. Beetz, K. Shea, G. Reinhart, K. Bender, C. Lau, M. Ostgathe, W. Vogl, M. Wiesbeck, M. E., C. Ertelt, T. Rühr, and M. Friedrich. *Changeable and Reconfigurable Manufacturing Systems*, chapter The Cognitive Factory, pages 355–371. Springer, 2009.

Manuel Giuliani received a Master of Arts in computational linguistics in 2004 and a Master of Science in Computer Science in 2006. He is currently working in the DFG-funded project AudiComm, which is part of the cluster of excellence "Cognition for Technical Systems" (CoTeSys). His research interests include cognitive robotics, human-robot interaction, natural language processing, multimodal fusion, and combinatory categorial grammar.

Claus Lenz received a Bachelor of Science degree in 2005 and a Diplom in 2007 in electrical engineering with a focus in human-machine communication. He is currently working in the research cluster "Cognition for Technical Systems" funded by the DFG in the excellence initiative on the project "Joint-Action for Humans and Industrial Robots" (JAHIR). His research interest include cognitive robotics, human-robot interaction and collaboration, and visual tracking.

Thomas Müller received a Diplom in Computer Science in 2007. He is currently working in the DFG-funded Collaborative Research Center SFB 453 on "High-Fidelity Telepresence and Teleaction". His research interests include human-robot interaction, bio-inspired vision systems, Lie algebras and screw theory for sensorimotor intergration, and parallelisation / virtualisation techniques.

Markus Rickert received a Diplom in Computer Science in 2004. He was working for the EU-funded project JAST "Joint-Action Science and Technology". His research interests include robotics, motion planning, human-robot interaction, cognitive systems, physics simulation, computer graphics, and software engineering.

Alois Knoll received his Ph.D. (summa cum laude) in computer science from the Technical University of Berlin, Germany, in 1988. He served on the faculty of the computer science department of TU Berlin until 1993, when he qualified for teaching computer science at a university (habilitation). He then joined the Technical Faculty of the University of Bielefeld, where he was a full professor and the director of the research group Technical Informatics until 2001. Between May 2001 and April 2004 he was a member of the board of directors of the Fraunhofer-Institute for Autonomous Intelligent Systems. At AIS he was head of the research group "Robotics Construction Kits", dedicated to research and development in the area of educational robotics. Since autumn 2001 he has been a professor of Computer Science at the Computer Science Department of the Technische Universität München. He is also on the board of directors of the Central Institute of Medical Technology at TUM (IMETUM-Garching); between April 2004 and March 2006 he was Executive Director of the Institute of Computer Science at TUM. His research interests include cognitive, medical and sensor-based robotics, multi-agent systems, data fusion, adaptive systems and multimedia information retrieval. In these fields he has published over 200 technical papers and guest-edited international journals.