Robotica (2007) volume 25, pp. 237–244. © 2007 Cambridge University Press
doi:10.1017/S0263574707003347 Printed in the United Kingdom

System-controlled user interaction within the service robotic control architecture MASSiVE

Oliver Prenzel†*, Christian Martens‡, Marco Cyriacks†, Chao Wang† and Axel Gräser†

†*Institute of Automation, University of Bremen, Bremen, Germany*
‡*Rheinmetall Defence Electronics GmbH, LTMR, Bremen, Germany*

(Received in Final Form: January 2, 2007)

SUMMARY
This paper presents an approach to reduce the technical complexity of a service robotic system by means of systematic and well-balanced user-involvement. By taking advantage of the user's cognitive capabilities during task execution, a technically manageable robotic system, which is able to execute tasks on a high level of abstraction reliably and robustly, emerges. For the realisation of this approach, the control architecture MASSiVE has been implemented, which is used for the control of the rehabilitation robot FRIEND II. It supports task execution on the basis of *a priori* defined and formally verified task-knowledge. This task-knowledge contains all possible sequences of operations as well as the symbolic representation of objects required for the execution of a specific task. The seamless integration of user interactions into this task-knowledge, in combination with MASSiVE's user-adapted human–machine interface layer, enables the system to deliberately interact with the user during run-time.

KEYWORDS: Semi-autonomy; Control architecture; Service robot; Verified task knowledge; Controlled human machine interaction; Human machine interface.

1. Introduction

The idea of a robotic assistant that supports people during daily life activities is as old as the history of robotics itself.[1] Even though this idea is still popular and challenging, its consequent realisation has to be stated as unrealistic at the moment: Assuming that solutions for currently unsolved technical problems, like the real-time interpretation of camera images, are at hand, a fully autonomous system is cost-intensive and performs tasks with poor efficiency and reliability.[2] This is because of its inherent high technical complexity. A manageable system with predictable behaviour is required. With respect to the current state-of-the-art, this requirement can be fulfilled only if the user's cognitive capabilities are taken into account, i.e. the robot executes the tasks semi-autonomously.[3–5]

The realisation of semi-autonomy is based on the central idea to include *human tasks* into task execution. A good explanation of this principle is given in the following quotation:

> *When we think of interfaces between human beings and computers, we usually assume that the human being is the one requesting that a task be completed, and the computer is completing the task and providing the results. What if this process were reversed and a computer program could ask a human being to perform a task and return the results?*[6]

With respect to a semi-autonomous service robotic system, this means that the user's cognitive capabilities are taken into account, whenever a robust and reliable technical solution is unavailable. It is obvious that the acceptance of such a system will be low in general. But, for people who are dependent on a personal assistant, like the disabled or elderly, this approach decreases their dependency and therefore increases their quality of life.

Beside a multitude of challenging technical tasks required for the realisation of a semi-autonomous system, the supply of an appropriate software-architecture plays a vital role.[7] Even though a number of implemented architectures is at hand, their development remains an active area of research in robotics.[8] In order not to cope with domain-specific overhead as well as with technical problems resulting from the modifications necessary to support the approach presented in this paper, we have been developing the control architecture MASSiVE (*M*ulti-layer *A*rchitecture for *S*emi-Autonomous *S*ervice-Robots w*i*th *V*erified Task-*E*xecution).[9] MASSiVE, designed for semi-autonomous rehabilitation-robots like FRIEND II[10], incorporates the dominating design principle of the "Three Tier Architecture"[11] and supports the required interaction between the user and the system, while executing tasks on the basis of pre-defined task-knowledge.

FRIEND II, as a possible representative service robotic system that is controlled with the concepts presented in this paper, consists of a 7-DOF (degrees of freedom) manipulator mounted on an electrical wheelchair. The gripper attached to the manipulator has an integrated force control and anti-slipping sensor. Further sensors that are used for task execution support are a stereo-camera-system with pan-tilt-heads, a smart tray (including a tactile skin to locate objects as well as a scale to measure their weight) and a force-torque-sensor installed at the robot's wrist. The presented hardware-setup of a rehabilitation robot is used

* Corresponding author. Email: prenzel@iat.uni-bremen.de

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

as a reference in the following, even though the application of MASSiVE is not limited to FRIEND II.

This paper is organised as follows: Section 2 gives an overview of the multi-layer architecture MASSiVE. MASSiVE offers an infrastructure for the incorporation of task planning (control on a higher level of abstraction) in connection with reactive behaviour (real-time reactions to dynamic environmental changes) as well as user interactions. The integration of task-knowledge-driven task execution with included user interactions is described in Section 3. It is shown how the user supports the acquisition of information required for task planning and task execution. Along with theoretical explanations, an exemplary interaction *identification of a cup* is derived. Section 4 focuses on the implementation of the human–machine layer, which realises task-context-dependent interactions. In Section 5, it is shown how to increase the functional capabilities of the system by system-controlled inclusion of respective user interactions without increasing its technical complexity. Finally, Section 6 gives a conclusion of the results achieved with the semi-autonomous approach and outlines further development steps.

2. Architecture

From the software technical point of view, the realisation of a service robot comes up with a multitude of challenging tasks to be solved within the context of a single architecture: distributed calculation, reactions to environmental changes and control of electro-mechanical devices in hard real-time, processing of complex algorithms, adaptation to heterogeneous hardware interfaces, ergonomic human–machine interaction or autonomous planning of action sequences. The preceding enumeration does not claim to be exhaustive. It turned out that hybrid multi-layer architectures, like TCA[12] or 3T[11], are predominating in the field of fully autonomous systems, since they provide a combination of deliberative and reactive behaviour. A deliberative component is necessary for the creation of a high-level plan, i.e. for having a system that is able to receive task requests and to define a mission goal. The inclusion of reactivity has proven to be a suitable mean to achieve robustness with respect to environmental disturbances, comparable to the reflex system of living organisms. Therefore, a typical hybrid multi-layer architecture consists of three layers:

- *Deliberator*: Plans operations on a high level of abstraction, i.e. with the help of a symbolic planning strategy.
- *Reactive Layer*: Has access to the actuators and sensors of the system and offers reactive operations in the form of closed control-loops (i.e. coupling of sensors and actuators).
- *Sequencer*: Is responsible for the activation and deactivation of operations in the reactive layer according to the plan generated by the deliberator. Therefore, it plays the role of a mediator between deliberator and reactive layer.

Besides these layers, a world model is included that contains the current system's perspective on the world according to the task to be executed. Due to the hybrid approach, a separation of world-model data into two

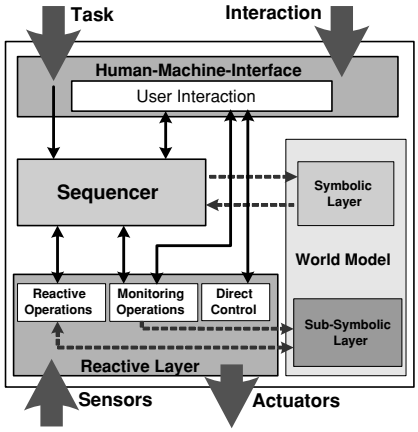


Fig. 1. Control-architecture MASSiVE.

categories is mandatory: The deliberator operates with *symbolic* object representations (e.g. *C* for the representation of a cup), while the reactive layer deals with the sensor percepts taken from these objects, so-called *sub-symbolic* information. Examples for sub-symbolic information are the colour, size, shape, location or weight of an object.

So far, the concept of hybrid multi-layer control architectures has been described generally. Even though it could be stated that MASSiVE belongs to this category, its structure differs slightly. Figure 1 depicts the emerging scheme of the architecture. With respect to the introductorily described assumption of semi-autonomy, the deliberator component of a traditional hybrid control architecture has been replaced by a human–machine interface (HMI), whereas symbolic planning is performed within the sequencer. The evident role of the HMI is the translation of high abstraction level task requests from the user (e.g. "*Fetch cup*", "*Pour in beverage*", etc.) into commands that are forwarded to the sequencer for further processing. Additionally, the HMI provides a complete infrastructure for task-related user interactions. This includes the direct control of actuators (indicated by the arrow between the HMI and reactive layer in Fig. 1) within context-based constraints given by the sequencer. The typical nature of user interaction context as well as the setup of the HMI infrastructure is the subject of Section 4.

Within MASSiVE, the role of the sequencer is to act as a global control unit that coordinates the cooperation between all levels of the architecture.[13] It is designed as a discrete-event-controller that operates on the basis of predefined task knowledge, so-called process-structures, and generates task-related action sequences. The usage of predefined task knowledge is justified by the application area of the service robot FRIEND II: In the domain of rehabilitation, frequently repeated tasks have to be processed under varying environmental conditions (e.g. serve beverage, prepare meals in microwave, reposition of objects, etc.). Despite their *a priori* definition, process-structures are still flexible enough to process a requested task under these different conditions, i.e. they can handle different initial situations and varying kinds of task participating objects. Due to the implicit restriction to task-relevant information, this approach guarantees, in contradiction to classical

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

AI-based task planning, a finite and, in the meaning of real-time suitability, reasonable size of the planning search space.

A planned action sequence consists of operations that are elementary from the sequencer's viewpoint. This motivates the term elementary executable operations (EEOPs). EEOPs include user interactions (e.g. identification of an object), direct control of actuators (e.g. movement of the camera system), monitoring operations (e.g. visual feature extraction of already identified object) and reactive operations (e.g. visually controlled object grasping). Due to the uniform software interface of EEOPs, the sequencer can generate action sequences independently of the kind of operation that has to be taken into account. After the generation of an EEOP-sequence is finished, the sequencer maps the EEOPs to so-called skills (i.e. quasi-parallel executable realisations of the associated operations) that are executed asynchronously on distributed software-servers. These servers are part of the reactive layer or the HMI, respectively. In the reactive layer, a network of servers offers basic system skills that are grouped within the servers according to functional togetherness. This structured and extensible infrastructure for skill-execution is implemented by means of CORBA. A further description of technical details can be found in ref. 9. However, some important features are summarised in the following:

- To enable parallel and physically distributed execution, re-parametrisation as well as the ability to abort execution skills are realised as asynchronous CORBA methods. To achieve real-time capability (e.g. for control loops), the CORBA communication overhead can be switched off by using the skill in the local process. This is comparable to a normal asynchronous function call and can be attained without any re-design of the system.
- Since the execution of EEOPs has been verified formally with respect to the supervision of the control and data flow as well as resource usage by analysing the content of the process-structures (see next section for details), the design of the reactive layer is slim and its complexity is consequently low.

So far, the architectural concept designed for task execution along with the structured inclusion of human tasks has been discussed. In the following, process-structures as a substantial aspect for the control of user interactions and autonomous system operations within MASSiVE are presented.

3. Task Knowledge-Driven User Interaction

According to the control architecture, process-structures come in two levels of abstraction: abstract process-structures PS_A are associated with the abstract (symbolic) layer of task knowledge representation. PS_A describe tasks in a user-oriented and non-technical form. This level is suitable for task-oriented programming purposes.[14] Elementary process-structures PS_E represent the task knowledge required in the environmental-related (sub-symbolic) level. The information on this level subsumes required hardware resources, like sensors and actuators, as well as EEOPs.

In the following, the different steps of task processing on the basis of abstract and elementary process-structures will be discussed. Figure 2 gives an overview on the steps that are undertaken to process a task request.

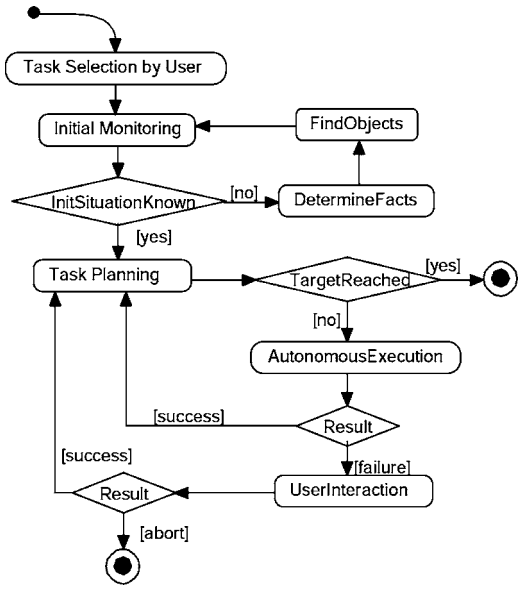


Fig. 2. Overview on task processing in MASSiVE.

3.1. Task request

First, with respect to the task request from the HMI, the sequencer selects and loads a PS_A. PS_A are derived from data-structures typically used in assembly planning, namely AND-OR-nets.[15] Within the context of MASSiVE they have been enhanced with first-order predicate logic facts that assign, in a STRIPS-like manner, pre- and post-conditions to the operations contained in the net (STRIPS: *S*tanford *R*esearch *I*nstitute *P*roblem *S*olver[16]). More specific, the facts have three purposes:

- Rejection of illogical states during offline-programming of PS_A. This results in a technically reasonable calculation complexity of the planning problem.
- Modelling of the state of the environment and system. This information is necessary during task execution as well as to establish a connection between the two layers of task knowledge abstraction.
- Control of the initial monitoring of the environment and determination of the initial situation in a PS_A, which is used as input for task planning.

Figure 3 depicts an iconic model of a possible PS_A that is assigned to the task request "*Fetch cup*". The iconic PS_A contains all object-constellations *OCs* (i.e. objects that are in physical contact) that might appear during the execution of this task. Within this context, *all* has to be interpreted as *from an application programmer's viewpoint*. The objects involved in the task scenario are a box B (e.g. an abstract representation of an already opened refrigerator), a cup C, the gripper of the service robot G and a table T. Some exemplary *OCs* of Fig. 3 are: OC_1 = "the cup is located inside the box", OC_4 = "the grasped cup is located inside the box" or OC_5 = "the cup is located on top of the table". On the basis of *OC*s, a situation S_k is described as follows: S_k is a set of *OCs*, that contains all objects of PS_A exactly once and is part of the

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

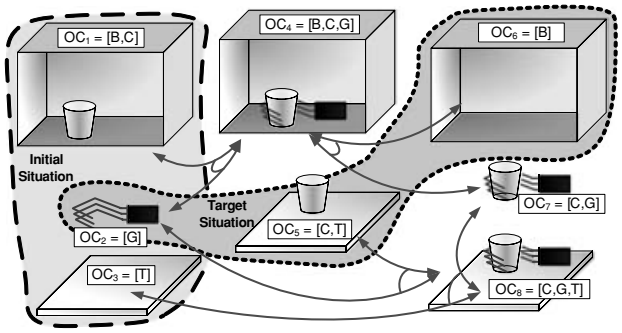


Fig. 3. Abstract process-structure for fetching a cup.

situation-graph[17], which interconnects all valid situations via the operations defined in PS_A. Thus, a target situation S_T can be reached via a finite set of operations, from any $S_k \neq S_T$. S_T has a fixed association with the task to be solved according to the given task request.

As depicted in Fig. 2 the initial monitoring starts after the scenario selection has taken place. Initial monitoring determines the situation of the chosen PS_A that corresponds to the current environmental situation. This initial situation is called S_I.

3.2. Initial monitoring

The situation monitoring process is controlled by the information contained in the set of situations associated with a PS_A. As illustrated in Fig. 2, the monitoring process consists of the determination of the first-order logic facts that are assigned to the different *OC*s as well as an object finding procedure, which is applied on the objects that appear as fact parameters. For a formal description of the semi-autonomous monitoring, see ref. 17. Within this context, the approach is explained by means of an example related to the given PS_A "*Fetch cup*".

Here, the fact $IsInside(C, B)$ has to be determined, describing the cup C being placed in the box B. Elementary process-structures PS_E are used for fact determination, since different sensors have to be accessed within the required monitoring operations. In the given case, the PS_E specifies the execution of monitoring skills, which acquire the location of the cup and box, and then decides about the fact's value in a subsequent calculation skill. Figure 4 depicts a simplified section of such a PS_E that specifies how to retrieve an object location either via autonomous operation on the machine vision server (upper part) or with the help of a user interaction (lower part). It is shown that PS_E are modelled with the help of Petri-nets. The advantage of using Petri-nets as a modelling tool is the availability

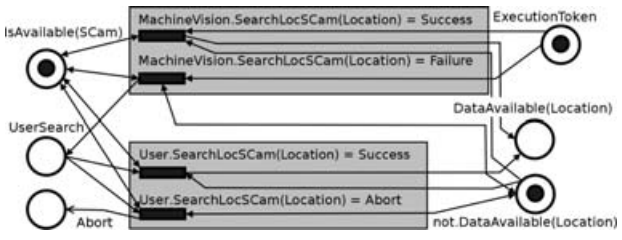


Fig. 4. Section of elementary process-structure.

of mathematical methods for the structural analysis of PS_E. Due to a stringent and application-specific syntax within a PS_E, offline verification becomes possible. This includes the verification with respect to dead-locks, resource-conflicts, modelling errors, state-reachability, availability of sub-symbolic data and the direction of their flow.[18] An example for the syntax used in the Petri-nets is the modelling of resource availability with an *IsAvailable*-predicate, as depicted in Fig. 4, where the *SCam*-item represents the stereo camera as system resource.

To determine a sequence of operations based on a process-structure, planning is necessary. The applied planning method is equivalent on both levels of process-structures (PS_A and PS_E) and is subject to the succeeding Section 3.3.

It is obvious that the complexity of the verification and planning task cannot be recognised within the depicted segment of a PS_E. This example has rather been chosen to illustrate the described approach. For a considerably more complex example, one can refer to ref. 18, where the placing of a meal into a microwave oven has been modelled, verified and simulated with the methods presented here.

Taking the example of finding a cup, the system will in most of the cases–and especially during initial monitoring–not be able to determine the location of a cup autonomously (i.e. the *MachineVision.SearchLocSCam* skill in Fig. 4 returns "*Failure*") and will therefore initiate a user interaction (*User.SearchLocSCam* skill) for the user-supported determination of the location within a camera image. A more concrete description of this situation is given with the help of experimental results in the following.

The FRIEND II system has been moved in front of a cupboard in which several cups are placed (Fig. 5). Within the mentioned user interaction, the user is requested by the system to control the pan-tilt head devices, so that the object of interest is in the sight of the camera (left camera image is displayed to the user). During the movement, the image processing system segments the objects positioned in the middle of the camera image in real-time. Within this application, the image's centre point is taken as seed point for a region growing algorithm,[19] in order to extract an image segment that represents the object of interest. The colour region including the seed point is growing in colour space by comparing the intensities in the three colour channels. The difficulty is to find a suitable colour range based on the seed pixel, which can be used to segment a certain area without any excessive or deficient growing. Considering this, an additional colour-edge extraction method has been used to stop the growing at the edges. Within the user interaction, the segmentation result is overlaid to the image, so that the user is able to decide whether the right object is highlighted or not.

To summarise the effect of the proposed semi-autonomous approach, it shall be compared to a more autonomous one within the same scenario setup (Fig. 5). According to the task to identify a cup for grasping, the system first has to determine, e.g. with the help of an *a priori* model, candidate objects for a cup. In case of a non-unique result, as in the given scenario, the system will not be able to finish the identification process successfully. Thus, the user has to be involved for a final determination of the target object, even though probably complex identification approaches have

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

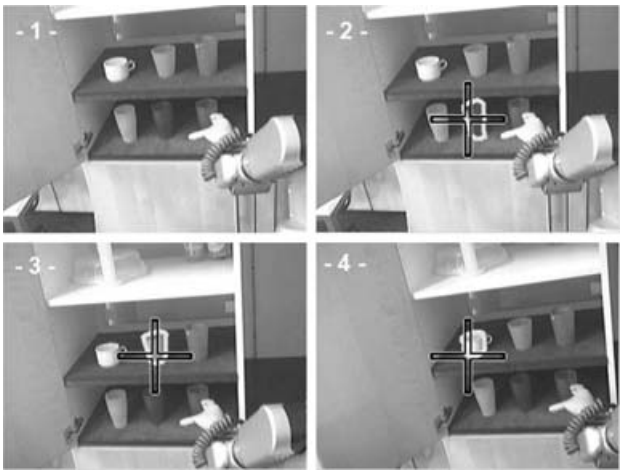


Fig. 5. Experimental results for finding a cup in a cupboard.

been used to determine candidate objects. Consequently, it is more effective to involve the user from the beginning on in conjunction with algorithms of lower complexity, like the proposed simple combination of segmentation and edge detection algorithms.

With respect to finishing the initial monitoring, the determination of other task-relevant objects is done analogously within a user interaction in case the autonomous object detection fails (or is not unique). After the initial object detection, additional object-specific sub-symbolic information is extracted according to the concept of object-anchoring.[17] This information is stored in the sub-symbolic layer of the system's world model (Fig. 1), to be accessible for all skills executed within the reactive layer. Thus, during further task processing steps, the degree of the system's autonomy increases with the increasing amount of acquired environmental information.

According to Fig. 2, task planning and execution takes place, once the monitoring of the initial situation S_I is finished for the loaded abstract process-structure PS_A.

3.3. Task planning and execution

In the given scenario with its assigned PS_A (Fig. 3) the task planning objective is to transform the initial situation $S_I = [OC_1, OC_2, OC_3]$ into the target situation $S_T = [OC_2, OC_5, OC_6]$. Since the transformation of situations is realised by means of the execution of operations and all possible operations within the current task scenario are represented as directed arcs connecting the OCs, the generation of action sequences is reduced to a graph search problem. Each operation of a PS_A is decomposed into an elementary process-structure PS_E (Fig. 4 already described a simplified section of such a process-structure), so that an initial sequence of operations to reach the mission goal can be generated by means of the Dijkstra Algorithm.[20] As illustrated by the activity diagram depicted in Fig. 2, autonomous skills are executed first and user interaction skills follow in case the system fails. This leads to a system that is able to operate robustly, even if it is not able to execute all steps in a fully autonomous manner. This approach enables an evolutionary increase of the autonomy level. Finally, the planning and execution cycle is continued until the given task is completed or has been aborted, e.g. according to the user's intention or due to a critical and unresolvable system status.

In case of the given sample scenario and due to the semi-autonomous determination of the cup's location within an initial user interaction, the next skill to be executed will be a grasping of the cup to fetch it from the cupboard. This means, a motion planning algorithm is started within a manipulative skill, which retrieves the relevant environmental information from the world model.[21]

For recapitulating this section, one can refer to Fig. 2. After the user's initial task selection, the first system-controlled user interaction takes places within the *initial monitoring*. Initial identification tasks that cannot be solved fully autonomous without unreasonable increase of technical complexity, are given back to the user to be solved within a human task. The same approach is taken into account within the ongoing process of task execution.

So far, the focus of discussion was dedicated to the task knowledge that drives the execution, respectively user interactions. It has not yet been discussed how the task knowledge is able to control and stimulate the user interactions. The succeeding section will answer this question.

4. Context-Aware Human–Machine Interaction

The part of MASSiVE that enables the inclusion of the user's cognitive capabilities during task execution is the HMI. It allows the user to start the execution of predefined system tasks (PS_A), like the previously described task "*Fetch cup*", on a high level of abstraction. Besides these functional aspects and according to the requirements of the rehabilitation domain, it provides a generic interface that maps the DOFs of any kind of input device, like eye-mouse or suck-blow-control, to the available system commands. For the realisation of the system initiated and controlled user involvement, the HMI offers an interface that enables the sequencing layer to initiate user assistance via user interaction skills. From the user-centred viewpoint, the overall objective is to offer an intuitively controllable interaction skill, which hides all technical details, like the selection of necessary hardware resources or the control of the data flow. In order to satisfy this requirement within an HMI that is flexible enough to be adapted to a variety of interaction skills, its implementation implies the design of a general and reusable structure for the interaction knowledge relevant to the HMI. This kind of structure is realised by means of the so-called *HMI-context*.

Before a definition of the *HMI-context* is given, the example presented in Section 3 is reconsidered to give an overview of the knowledge belonging to the *HMI-context*. Within this example, the user is consulted by the sequencer to assist the system locating a cup by a call of the user interaction skill *User.SearchLocSCam*. For its execution, a list of available hardware resources, participating environmental objects as well as their states and relations is required. Furthermore, verbal user instructions describing the interaction request are necessary. All this information is stored within the symbolic part of the world model. During the instantiation of a user interaction skill, this information is addressed by the actual skill parameters and

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press



Fig. 6. The user interaction dialogue.

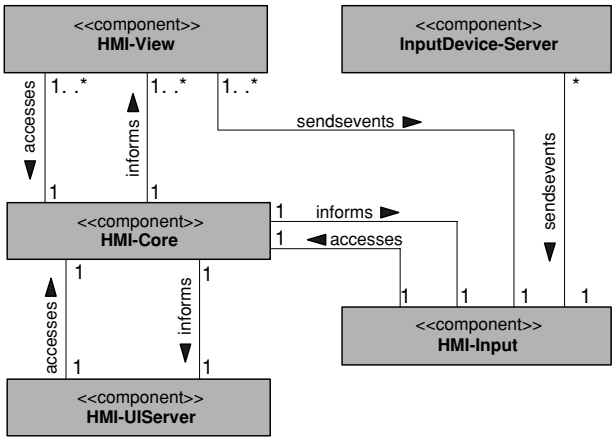


Fig. 7. The components of the HMI.

stored within the *HMI-context*. According to the example of locating a cup, the dialogue setup resulting from the associated *HMI-context* is depicted in Fig. 6. Within this context, the dialogue consists of a navigation area for the camera's pan-tilt-heads (hardware resources) and the current camera-view with segmentation results, which the user has to associate with the cup the system is looking for (environmental object).

To ease the development of user interaction skills, all possible components of user interaction dialogues are combined in the form of a component toolbox. There are two different kinds of such components: Data consumers and producers. The consumer component is used for display purposes only. Components of that kind read data from the *HMI-context*. Producer components generate data, like user input data resulting from cursor movements or button selections. This data is stored within the *HMI-context*. To enable a handicapped user to process his or her *human task*, each component of the latter kind stores information about the command context. This information is used to support the context-dependent mapping of the input devices' DOFs, so that it is possible to map the incoming events from input devices to the currently available commands.

Besides the data used for user interaction skills, there is the necessity to store additional information concerning the main user interface of the HMI. Since this data is almost similar to the data described before, no further description is stated here. These considerations lead to the following set of elements included in the *HMI-context*:

- hardware resources for the *human task*
- environmental objects and their states/relationships
- verbal explanations and instructions for the user
- components of the user interaction dialogue
- data consumed or generated by components of the user interaction dialogue including possible user-commands
- components and data of main user interface

Thus, the *HMI-context* is a subset of elements that conforms with the more general definition of the term context to be found in ref. 22:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.*

For the realisation of the *context-aware* behaviour, the HMI consists of several components associated with specific roles. The design of the connection between these components is based on the observer design pattern.[23] As depicted in Fig. 7, the observer design pattern in case of the HMI describes a one-to-many, respectively a one-to-one dependency between its components. Here, the HMI-Core, which acts as *subject*, manages the *HMI-context*, so that it is possible to decouple the context-dependent knowledge from the remaining part of the HMI, especially the input devices. The HMI-Core is bidirectionally connected to its *observers*: HMI-View, HMI-Input and HMI-UIServer. If the context changes, the *observers* are notified via the HMI-Core to reload the data relevant for the new context. Beside this context-management, the HMI-Core component is responsible for command interpretation and the resulting changes in the context that are described as follows.

The HMI-View component represents the graphical user interface and is the only component visible to the user. The HMI-Input component realises the generic adaptation of different input devices. It handles the events produced by so-called InputDevice-Servers that are implemented for each physical input device. In a first step, all events coming from these InputDevice-Servers are mapped to one of three event kinds: *cursor move event*, *cursor click event* or *command*. The resulting events are sent to the HMI-Input component, which is responsible for mapping them to associated commands within the current *HMI-context*. Afterwards, the mapped commands are forwarded to the HMI-Core to cause a change of the current context, if necessary. As depicted in Fig. 7, the HMI-View component acts, according to the HMI-Input component, as a InputDevice-Server. This is necessary, if events coming from ordinary input devices (e.g. mouse and keyboard) have to be handled in the same manner as events emitted from special input devices, without bypassing the operating system.

The connection between the HMI and MASSiVE is established by the HMI-UIServer component. It uses the interface of the sequencer to start and stop system tasks and offers a skill-server to enable the sequencer initiating user

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

interactions. So, a user interaction is, *from the sequencer's point of view*, nothing else than some kind of EEOP.

5. Human Task-Based Increase of System Capability

The task planning process described in Section 3 on the basis of the sample process-structure of Fig. 3 will lead to the necessity to put down the cup onto a table (assembly of OC_7 and OC_3 to gain OC_8). This, in turn, includes a monitoring operation to find a suitable table or equivalent platform with enough free space on it to place the cup. In typical service robotic environments, it is a challenging task to automate this procedure of finding even platforms for placing objects. Usually, different sensor systems have to be used, sensor data has to be fused and complex algorithms are to be applied. Finally, due to different and unstable environmental conditions (e.g. light conditions, obstacles, complex surface textures or properties, etc.) the system can still not be 100% sure to do the job right after autonomous detection of the required platform place. Therefore, in the case of a service robot for an elderly or disabled person who surveilles the task execution, the inclusion of a human task into the ongoing task execution is reasonable and increases the system capability with respect to many aspects. Finding a free place to put down an object is just one of many sample human support tasks that are candidates to be considered. The approach presented in this paper provides the infrastructure for this human task inclusion. The mentioned user interaction of finding a free platform place will be illustrated with experimental results in the following.

Again, the same combination of segmentation and edge detection from Section 3.2 is applied. Figure 8 depicts the snapshots from the live view of the left camera shown to the user. Here, determined by a slightly different user interaction context, the user is able to move a cursor in the camera image and the current position of the cursor is taken as seed point for the region growing.

It can be seen again that the combination of the two algorithms of lower complexity within a user-interaction is suitable to identify homogeneous regions, which can be assigned as free and even platform regions by the user. Within the depicted snapshots, the cursor is moved and the segmentation algorithm first highlights nothing (1), then the microwave-oven's door (2), the door of the lower closed cupboard (3) and finally a free place to put down the cup on the right side (4). The respective dialogue for the user interaction is–beside the displayed camera view–identical to the one depicted in Fig. 6.



Fig. 8. Experimental results of finding a free platform position.

6. Conclusion

In this paper, the MASSiVE approach of controlled user involvement into task execution with a service robotic system has been presented. The special contribution of MASSiVE is to make use of predefined task-knowledge that is still flexible enough to be adjusted dynamically to changing environmental conditions during task execution. Moreover, the *a priori* defined and verified task-knowledge is utilised to adapt a context-sensitive human-machine-interface to the circumstances of a certain user interaction. This HMI provides an infrastructure that hides non-relevant information from the user, gives clear input requests, coordinates the user interaction within interaction-specific constraints (e.g. with respect to the resources to be used) and finally assures the availability of requested information as well as its conformity to the overall process of task execution. The application of the presented methods has been demonstrated with the help of two representative sample user interactions, namely "*identify a desired object*" and "*specify a free place to put down an object*".

Within the next development steps, the transfer of further user interactions will take place. For instance, it is required to enhance the ability to extract relevant 3D-information of objects (i.e. size and location) from sensor percepts. This means, the extraction has to work, e.g. on a cupboard or fridge as well as on a worktop or the system-tray. From the viewpoint of the current state-of-the-art, the fulfilment of this requirement can be judged as realistic only, if the user is involved into the process of object localisation. Especially, the integration of operations that have not yet been realised as autonomous operation (e.g. no technical solution available, lack of real-time suitability, etc.) becomes possible due to the interaction with the user. Subsequently, MASSiVE's infrastructure will be used to increase FRIEND II's autonomy in an evolutionary manner. Hence, the MASSiVE concept provides a significant impact for the realisation of a large number of daily-life rehabilitation scenarios within the near future.

References

1. J. F. Engelberger, *Robotics in Service*, 1st ed. (MIT Press, Cambridge, MA, 1989) ISBN 0-2620-5042-0.
2. P. Dario, R. Dillman and H. I. Christensen, "Euron Research Roadmaps." Key Area 1 on "Research Coordination." Available: http://www.euron.org (2004)
3. C. Laschi, G. Teti, G. Tamburrini, E. Datteri and P. Dario, "Adaptable Semi-Autonomy in Personal Robots," *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication*, Bordeaux, Paris, France (2001) vol. 10, pp. 152–157.
4. C. Martens, D.-J. Kim, J.-S. Han, A. Gräser and Z. Z. Bien, "Concept for a Modified Hybrid Multi-Layer Control-Architecture for Rehabilitation Robots," *Proceedings of the 3rd International Workshop on Human-Friendly Robotic Systems*, (Jan. 21–22, 2002), pp. 49–54.
5. E. Colle, Y. Rybawzyk and P. Hoppenot, "Arph: An Assistant Robot for Disabled People," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*,

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press

Hammanet, Tunisia (2002) vol. 1, pp. 176–181. ISBN 0-7803-7437-1.
6. Amazon.com, "Amazon Mechanical Turk—Artificial, Artificial Intelligence." URL. Available: http://www.mturk.com (2006).
7. M. Hans, *Eine Modulare Kontrollarchitektur für den Hol- und Bringdienst von Roboterassistenten Ph.D. Thesis* (Germany, Stuttgart: University of Stuttgart, 2005).
8. I. A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I.-H. Shu and D. Apfelbaum, "Challenges and steps toward reusable robotic software," *Int. J. Adv. Robot. Sys.* (2006). **3**, 23–30 (2006), ISSN 17298806.
9. C. Martens, O. Prenzel, J. Feuser and A. Gräser, "MASSiVE: Multi-Layer Architecture for Semi-Autonomous Service-Robots with Verified Task Execution," *Proceedings of the 10th International Conference on Optimization of Electrical and Electronic Equipments*, Brasov, Romania (2006) vol. 3, pp. 107–112, ISBN 9-7365-3705-8.
10. O. Ivlev, C. Martens and A. Graeser, "Rehabilitation Robots FRIEND-I and FRIEND-II with the Dexterous Lightweight Manipulator," *Proceedings of the 3rd International Congress: Restoration of (Wheeled) Mobility in SCI the Rehabilitation*, Amsterdam, The Netherlands (Apr. 19–21, 2005), vol. 5, pp. 111–123.
11. R. Bonasso, D. Kortenkamp, D. Schreckenghost and D. Ryan, "Three Tier Architecture for Controlling Space Life Support Systems," *Proceedings of the IEEE SIS'98*, Rockville, MD, USA (May 21–23, 1998), pp. 195–201.
12. E. Coste-Maniere and R. Simmons, "Architecture, the Backbone of Robotic Systems," *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, USA (Apr. 2000), vol. 1, pp. 67–72. ISBN 0-7803-5886-4.
13. C. Martens and A. Gräser, "Design and Implementation of a Discrete Event Controller for High-Level Command Control of Rehabilitation Robotic Systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland (Sep. 30–Oct. 4, 2002), pp. 1457–1462.
14. C. Martens, "Task oriented programming of service-robots on the basis of process-structures," *Methods Appl. Autom.*, 45–56 (2005), ISBN 3-8322-4502-2.
15. T. Cao and A. C. Sanderson, "AND/OR net representation for robotic task sequence planning," *IEEE Trans. Syst. Man Cybern.* **28**, 204–218 (1998).
16. R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.* **2**, 189–208 (1971).
17. O. Prenzel, "Semi-autonomous object anchoring for service-robots," *Methods Appl. Autom.*, 57–68 (2005), ISBN 3-8322-4502-2.
18. C. Martens, *Teilautonome Aufgabenbearbeitung bei Rehabilitationsrobotern mit Manipulator—Konzeption und Realisierung eines Softwaretechnischen und Algorithmischen Rahmenwerks Ph.D. Thesis*, Bremen, Germany (University of Bremen, 2004). Shaker Verlag.
19. Zucker, S. W., "Region Growing: Childhood and Adolescence," *Computer Graphics and Image Processing* **5** (3) 382–399, 1976.
20. C. Martens, "Generation of parallel executable control sequences for rehabilitation robotic systems on the basis of hierachical petri-net based task representation," *Methods Appl. Autom.*, 73–85 (2003).
21. D. Ojdanic, O. Ivlev and A. Gräser, "A new fast motion planning approach for dexterous manipulators in 3D-Cartesian space," *Proceedings of the ISR-Robotik Joint Conference on Robotics*, Munich, Germany (May 15–17, 2006) p. 49.
22. A. K. Dey, G. D. Abowd: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the Computer-Human Interaction 2000 (CHI 2000), Workshop on The What, Who, Where, When, and How of Context-Awareness, The Hague, The Netherlands, April 2000.
23. E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software* 1st ed. (Addison-Wesley, Reading, MA, 1995). ISBN 0-2016-3361-2.

https://doi.org/10.1017/S0263574707003347 Published online by Cambridge University Press