IEICE TRANS. INF. & SYST., VOL.E93–D, NO.6 JUNE 2010

INVITED PAPER *Special Section on Info-Plosion*

A Framework for Information Distribution, Task Execution and Decision Making in Multi-Robot Systems

Matthias RAMBOW[†a)], **Florian ROHRMÜLLER**[†], **Omiros KOURAKOS**[†], **Dražen BRŠČIĆ**[†], **Dirk WOLLHERR**[†], **Sandra HIRCHE**[†], ***and*** **Martin BUSS**[†], ***Nonmembers***

SUMMARY Robotic systems operating in the real-world have to cope with unforeseen events by determining appropriate decisions based on noisy or partial knowledge. In this respect high functional robots are equipped with many sensors and actuators and run multiple processing modules in parallel. The resulting complexity is even further increased in case of cooperative multi-robot systems, since mechanisms for joint operation are needed. In this paper a complete and modular framework that handles this complexity in multi-robot systems is presented. It provides efficient exchange of generated data as well as a generic scheme for task execution and robot coordination.

key words: *multi-robot system, robotic architecture, autonomous robots*

1. Introduction

One of the most challenging problems in robotics is to operate robustly in real-world environments. To this end, the robot has to cope with both unexpected, previously not planned situations - often referred to as surprise - and unavailable data due to sensor failure, either through environment conditions or due to hardware defects. Such abilities are commonly subsumed under the term "cognitive". A common way to implement such cognitive systems is to integrate a large number of sensors of different modalities, fuse their measurements to a concise representation and on the other hand implement a large number of modules with different abilities and functionalities, who are highly dependent on their mutual results. This strong interdependency of modules and sensors creates highly complex systems that are difficult to maintain from an engineering perspective and impose major challenges on handling the data flow within the system.

In the case where the tasks that need to be executed require joint operation of multiple autonomous robots, there is further a need for mechanisms for inter-robot coordination and data sharing. These requirements in multi-robot systems call for the application of a general framework that is concise and modular, in order to keep the complexity and amount of information in systems of multiple robots easily manageable.

The perception of the surroundings is one of the main requirements for autonomous operation. In order to obtain an extensive representation of the environment while being robust to disturbances and surprise, cognitive robotic systems use multiple sensors with different sensing modalities, whose data are processed and fused. An example of a robotic system with complex perception is BOSS [1], the winner of the 2007 DARPA Urban Challenge [2], which used a combination of ten laser range finders, five radars, two high-dynamic-range cameras and one GPS sensor. In multi-robot systems the number of sensors grows proportionally with the number of robots, and it is often necessary to exchange the perception data between robots. In order to cope with the large amount of information, the processing and exchange needs to be done on several levels of abstraction, i.e. exchanging pre-processed data that contains only the relevant information rather than exchanging raw sensor data. This allows for efficient data exchange and for a good tradeoff between the fidelity of information and the amount of transferred data.

While it is beneficial to exchange perception data between multiple robots, the execution of tasks is performed separately on each robot. Modern autonomous robots are typically able to perform multiple tasks and often also several tasks in parallel. Their execution needs to be both efficient and stable, hence the communication between control modules needs to be fast. Furthermore actuator commands need to be definite and therefore should be computed at a central location, i.e. single robot control requires centralized data processing.

Most of the examples of advanced autonomous robots today use a framework with a structure that includes the just described parts – multi-layered data processing and centralized task execution. In addition, they typically utilize some middleware solution, which shows the importance of an appropriate data exchange mechanism. Junior [3], the runner-up at the DARPA Urban Challenge, used a software architecture that is organized in groups of modules connected in a pipeline structure. The data from the sensor interface are routed to the perception group that extracts moving and static obstacles and localization data. These are used by the navigation group, which determines the vehicle's behavior. The humanoid robot ARMAR-III [4] uses the MCA framework [5] for efficient communication between software modules, whereas for the control of the robot a hierarchical architecture divided into three layers is used. The task planning layer specifies subtasks for the different kinematic subsystems, i.e. head, arms, hands, etc. The task coordination layer is responsible for activating sequential and/or parallel actions of the subsystems, which are then in turn

Manuscript received February 26, 2010.

[†]The authors are with the Institute of Automatic Control Engineering (LSR) at the Technische Universität München, Germany.

a) E-mail: rambow@tum.de

DOI: 10.1587/transinf.E93.D.1352

executed by the task execution layer. Okada et al. propose in [6] an integrated software architecture for the humanoid HRP2. The low level part of it is the HRP2 control system [7] responsible for hardware interfacing, whereas the higher system level functionality is centrally implemented using EusLisp [8]. The two parts are connected using the CORBA architecture [9].

The above frameworks are suited well for controlling one complex autonomous robotic agent, however they do not incorporate tools to deal with multiple robots. To obtain collaboration among multiple robots respective planning and coordination is required. This can be achieved either via explicit robot-robot communication, such as in [10]–[12], or by fully distributed decision making, i.e. modeling and observing the other robots such as in [13], [14]. The communicational complexity resulting from a full mutual message exchange grows exponentially with the number of robots, but without communication the solution quality is suboptimal and strongly dependent on the perceptual capabilities and the accuracy of the models. Since the number of robots in real-world settings is usually rather small a communicative solution is in general preferable.

Nevertheless also for such systems various degrees of centralization are possible. One approach is a centralized decision maker, which computes the best plan for the entire system and forwards its solution to all robots. Such an approach provides poor robustness due to a single point of failure and is unfavorable in terms of computational complexity [15]. Especially when the number of simultaneously upcoming tasks is large, the latter is problematic. For this reason, instead of using rigid centralized decision making, it is advantageous to perform multi-robot planning and coordination in a distributed manner. Consequently, a complete framework for multi-robot systems needs to have a hybrid centralized-distributed structure.

The approach presented in this paper is to implement such a hybrid framework for systems of multiple heterogeneous autonomous robots, which smoothly integrates the handling of both real-time control requirements on the single robot and information exchange for collaborative multi-robot scenarios. This is done using mainly existing state of the art frameworks and middleware solutions, which are integrated into a concise and modular framework. We describe a complete and generally applicable approach, starting from the communication framework to the multi-layered structure of multi-robot task execution and coordination.

The paper is organized as follows. The layered structure of the proposed multi-robot framework is introduced in Sect. 2. A communication mechanism, which is used to connect the various data streams and processing modules, is described in Sect. 3. In Sect. 4 we present our coordination and execution architecture for organizing and structuring the various modules in different abstraction layers. An example implementation, which emphasizes the motivation for this work, is presented in Sect. 5.

2. Framework Structure

Figure 1 shows the structure of our proposed framework, which is running on each robot in the multi-robot system.

On the bottom is the robot hardware, which includes sensors and actuators. Here raw measurement data are produced and the actuators commands are executed. The Intra-Robot Processing part in the middle is responsible for the execution of tasks that are assigned to the robot, and comprises the processing of sensor data and calculation of actuator commands. In contrast to Intra-Robot Processing, which deals only with the data processing inside one single robot, Distributed Processing, the top part of the framework, is responsible for making decisions on the level of the complete multi-robot system. Its main function is to find the appropriate assignment of actions to the robots, and as mentioned before, it is implemented in a distributed manner.

The data transportation inside and between the described parts, as well as the communication between the robots, is enabled by a middleware. It encapsulates the communication and provides interfaces to the hardware and thereby helps keeping programming efforts low.

The details of these framework parts are given in the following sections.

3. The ARCADE Middleware

All the functions necessary for the system operation, such as data processing or control methods, are implemented in the framework in separate software modules. The modular structure makes the framework more flexible and enables easy development, maintenance, and extension of the system functionality.

The necessary connections between the modules are provided by a communication architecture, commonly referred to as middleware. It abstracts the communication between the modules by providing standard interfaces through which data can be exchanged.

Our multi-robot centralized-distributed framework poses several requirements on the middleware. In order to

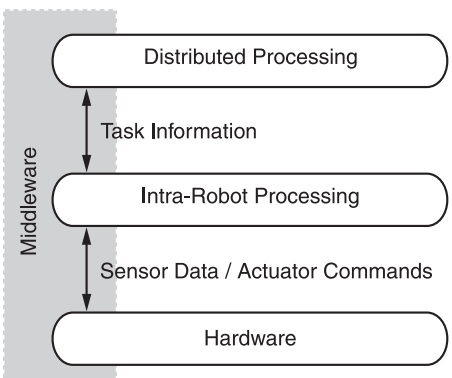


Fig. 1 Overall structure of the proposed framework.

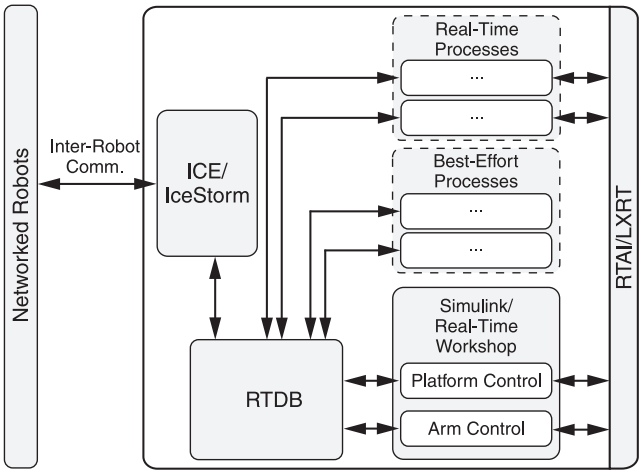


Fig. 2 Overview of the *ARCADE* middleware.

enable the centralized high frequency control within each robot, fast intra-robot communication needs to be possible. On the other hand, efficient communication between the robots must be provided.

In order to fully satisfy these requirements, we utilize *ARCADE* (**A**rchitecture for **R**eal-time **C**ontrol and **A**utonomous **D**istributed **E**xecution) [20]. It both supports real-time intra-robot data exchange and provides efficient distributed communication.

Examples of other popular middleware solutions for robotic applications are *Player* [16], *ROS* [17], *YARP* [18], and *OROCOS* [19]. *OROCOS* is the only inherently real-time capable among them. However, the distributed communication mechanism of *ARCADE*, which is based on ZeroC/ICE [21], has better performance and is easier to use than the one of *OROCOS*, which uses CORBA [9].

The structure of the *ARCADE* middleware is shown in Fig. 2. The implementation of intra-robot and distributed communication is described next. Further details on *ARCADE* can be found in [20].

3.1 Local Module Communication

A major advantage of *ARCADE* is the local inter-process communication that provides seamless data acquisition and sharing among multiple modules in one computer even under real-time constraints. The mechanism used for that is the real-time database *KogMo-RTDB* [22], which, although not a database in the traditional sense, serves as a central data repository where information can be stored and fetched. However, in comparison to other similar data exchange schemes, RTDB has excellent timing characteristics. Its average/worst case times in a strongly busy system with real-time configuration are $23\,\mu s/134\,\mu s$ for write operations and $17\,\mu s/62\,\mu s$ for read operations respectively [23]. In addition, it provides several other useful features, such as record/replay of states, buffering of data, etc.

This allows *ARCADE* to be used for the modular implementation of high-frequency centralized control, while meeting real-time constraints.

3.2 Distributed Communication

For connecting modules running on different robots a different mechanism for communication is required, which is achieved by transferring data among the respective RTDBs. For this purpose ZeroC/ICE [21] is integrated into the *ARCADE* middleware. Through ICE, as well as the IceStorm multicast extension, access and data transfer between all RTDBs within the entire system is possible. This allows any two modules in the multi-robot system to efficiently exchange data in the same way as modules running on the same robot, although in this case the timing guarantees are not as strict.

4. Information Processing in Cooperative Multi-Robot Systems

The objective of a robotic system is to perform dedicated tasks in an efficient and autonomous manner. To achieve this objective it needs to be able to:

1. extract information from raw sensor data and create a model of the environment,
2. use this model to make appropriate decisions,
3. execute these decisions efficiently,
4. handle failures or performance drops.

While all these issues have to be addressed for single-robot systems, for multi-robot systems extended solutions are required in order to reach a system-wide consensus based on information dispersed among different robots.

For the achievement of the above requirements, modules for perception, planning and actuator control are necessary. These permanently produce and exchange a large amount of new information. In order to coordinate the interactions of these modules in complex distributed systems and to enable the cooperative and cognitive operation the proposed framework uses the scheme shown in Fig. 3. It consists of four main blocks:

- The Information Extraction that is responsible for processing sensor data and extracting information at different abstraction levels.
- The Task Execution that uses the information to generate actuator commands that achieve a desired change in the world.
- The Task Allocation that is responsible for distributed decision making.
- The Performance Control that provides the connection between the distributed Task Allocation and the centralized Task Execution.

In the following sections each block is explained in more detail.

4.1 Information Extraction

Information Extraction comprises all modules within the

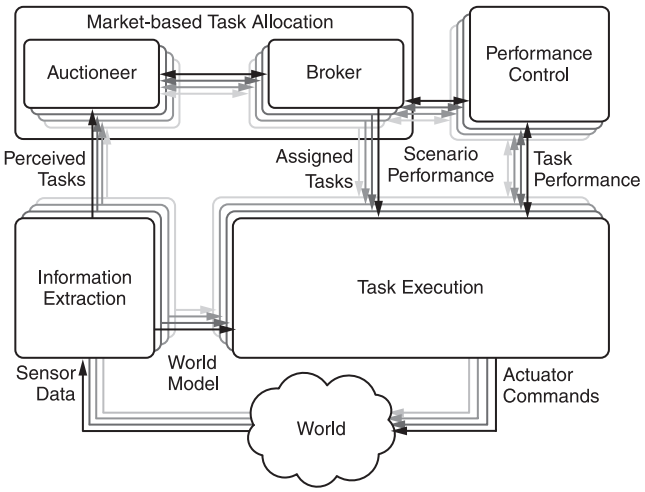


Fig. 3 Main functional blocks of the multi-robot framework and their interconnections. Layers with different shades of gray denote instances on different robotic agents in the system.

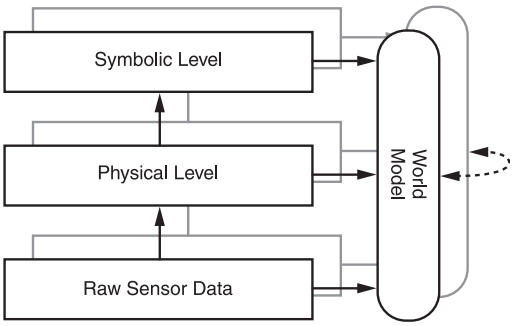


Fig. 4 Abstraction levels of the information gathered by the Information extraction module.

system that transform lower level input data into some higher level information and thereby create a world model that represents the current state of the surrounding environment. As shown in Fig. 4, information is gradually processed from raw sensor data to physical information and further to symbolic high level knowledge. The information from all the different levels makes up the world model.

The different levels of abstraction provide the means to optimally make use of the available information not only locally but also for the information exchange between robots. If a robot needs information gathered by another robot, it requests this information from the abstraction level that is most suitable for the task completion and integrates it in its own world model. In Fig. 4 this optional data request is indicated by the dashed arrow. This keeps the complexity low and conserves the potentially limited computational and communicational resources.

4.2 Parallel Execution of Tasks

Complex robotic systems are able to execute more than one task in parallel. Here a task is in essence a mapping from the current information about the world state to an actuator command that will cause a desired change in the world. By appropriately combining several simpler tasks it is possible to realize tasks of larger complexity. In order to do that mechanisms for scheduling and prioritization of tasks, and merging of possibly conflicting commands from tasks that run in parallel need to be implemented. This is the responsibility of the Task Execution.

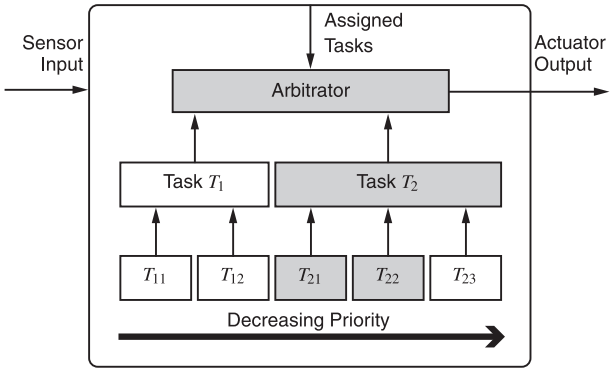


Fig. 5 Example of a hierarchical task tree. The (needed) sensor inputs are forwarded top-down, the actuator outputs are forwarded and combined bottom-up. Grey color indicates active tasks.

In our framework a hierarchical task tree structure as shown in Fig. 5 is used, which allows the realization of parallel and sequential combinations of tasks and transitions between them. Tasks on higher levels of the task hierarchy are more complex and can be decomposed into a set of subtasks, e.g. in the figure T_{11} and T_{12} are subtasks of task T_1. This hierarchical decomposition allows for reusability and modularized implementation of subtasks.

The leaf nodes of the tree are primitive tasks that encapsulate the actual execution modules, e.g. motion primitives of the robot or data processors. Each primitive task calculates some output data based on the current state, input data and a set of parameters.

Subtasks have defined priorities which specify how their outputs are fused. Each parent task is responsible for combining the outputs of its subtasks it is controlling and forwarding the combined output to its own parent. In case multiple subtasks are active and use the same hardware resource, contradictory commands have to be detected and handled. A common approach for this problem is fusing the actuator commands by projecting lower priority subtasks into the null-space of higher priority subtasks [24].

In order to define the scheduling, i.e. time sequencing, of subtasks, all of the tasks include a finite state machine. This allows the combination of subtasks in a parallel and/or sequential manner. Switching of states is triggered either by active subtasks or by perceived changes in the world model.

The arbitrator is the center of the task tree and the central element of the Task Execution. It is responsible for combining the control outputs of activated high level tasks as described above and sending the control commands to the robot actuators. It also controls and routes the information flow among the internal execution modules and to the plan-

ning modules. Furthermore, it provides the interfaces to the planning modules described in the next section, and activates and deactivates tasks in the second level of the task tree according to the received task assignments.

This centralized design provides the coherent control of a kinematic system as it is given by the physically connected hardware of a robot. Since robots are not kinematically connected among each other, except for cooperative tasks where physical interaction is required, a separate execution control for each robot is in general sufficient. However, to yield a cooperative system behavior a joint decision making is required.

4.3 Decision Making in Distributed Systems

While the task execution provides only the realization of tasks, a robot requires also planning modules that identify adequate tasks with respect to the current situation. In case of multi-robot systems multiple tasks for multiple robots need to be determined. Commonly the aim in multi-robot systems is that the robots coordinate their tasks and yield a cooperative behavior. This provides advantages such as speedup through parallel execution or increased robustness due to redundancy [15]. The essential problem to be solved is deciding which robot should perform which task, commonly known as task allocation problem.

In our framework we use a market-based approach [10], which provides a good trade-off between a fully decentralized and a fully centralized system design. Market-based task allocation follows the idea of economical markets. The principle used in our framework is illustrated in Fig. 6. For each new task there is one auctioneer which is responsible for the announcement of the tasks, whereas multiple brokers bid for the respective assignment. In general each robot in the system runs one broker and one auctioneer. An auctioneer retrieves the new tasks from the information extraction on its host robot. The brokers compute bids in form of performance metrics, i.e. the cost or the profit incurred by the task, and reply it to the auctioneer. After receiving bids from all brokers, the auctioneer makes a locally centralized decision in order to optimize the global system performance metric. Thereby the optimality of the centralized design is combined with the robustness of the decentralized one, since any robot in the system can act as an auctioneer. The computation of bids on each robot depends on several factors and is discussed next.

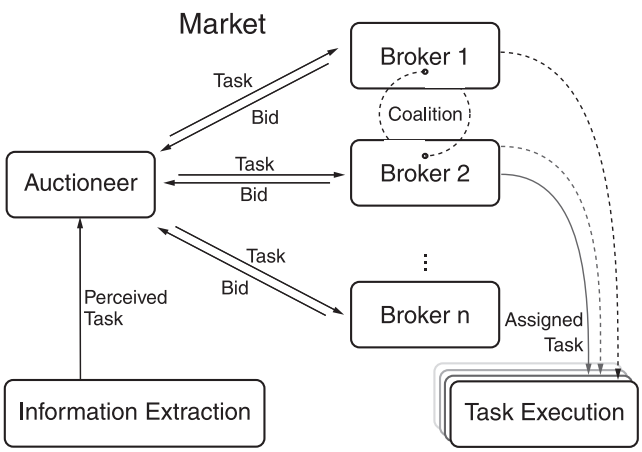


Fig. 6 The principle of market-based task allocation: The auctioneer forwards perceived tasks to the brokers. These compute their respective bids based on which the auctioneer assigns the task to the most efficient broker or coalition (team). In the example case here, Broker 1 and 2 form a coalition for one task while Broker 2 executes a further task alone. No task is assigned to Broker n.

4.4 Performance Control in Complex Systems

As stated in the previous section, in order to assign the tasks an auctioneer performs the optimization of a global, scenario specific performance metric. The brokers running on the robots therefore need to express their bids in this same metric, which has to be calculated based on local performance metrics of tasks and subtasks. However, in general these local subtask metrics differ from the global one, making the calculation of the bid not straightforward. As an example, a control subtask for the robot hand might be designed to yield trajectories with minimum jerk, while a collision avoidance subtask could aim to generate trajectories with maximum clearance. On the other hand, the overall global objective of the entire system may be entirely different and could be for example minimizing the time required to serve a drink to a person.

Due to the interconnection of the tasks, the respective performance metrics are also interdependent. In order to be able to express the performance in the global metric, these interdependencies between the metrics needs to be determined.

Deterministic relations between the task metrics and the global metric might be hard to identify or too complex to be modeled. Therefore we utilize the system interdependence analysis described in [25], where the metric interdependencies are modeled probabilistically by searching and training a Bayesian Network structure. The Bayesian Network enables the quantification of the mutual metric interdependencies whereby the relation between task-specific and scenario-specific metrics is obtained. This enables a performance estimation on all layers – i.e. from low-level control up to high-level decision making – that takes the system interconnectivity into account.

By this system-wide performance consideration a tight coupling of the previously presented task allocation, where the objective is to optimize the scenario performance, and task execution, where the task performance is optimized, is achieved. In combination with the *ARCADE* middleware an all-encompassing framework is given that enables the cooperative operation of multiple complex robots. An implementation example is presented next.

5. Experimental Application

The presented approach has been integrated on our multi-

robot system and tested in a service scenario. First, the used hardware setup is described. Then, the cooperative task is briefly introduced and subsequently the timeline with the centralized execution and distributed decision making is presented.

5.1 Hardware Setup

Figure 7 shows one of the two robots involved in this example scenario. Both robots, labeled as R1 and R3, have a four-wheeled omnidirectional mobile platform [26], which offers human-like maneuverability and smooth motions. Two identical anthropomorphic 7-degrees-of-freedom (DoF) arms are front-mounted on the top of the main chassis to provide a human-like working space [27]. R3, shown in Fig. 7 is equipped with a Schunk PG70 two-finger gripper that is attached to the right arm. The second one, R1, uses a three-fingered BarrettHand for more dexterous manipulation.

For safe navigation each robot has two Sick S300 laser range finders placed in opposing corners of the chassis to allow for circumferential planar obstacle detection during navigation. On top of R3 there is the emotion display head EDDIE [28], which enables dynamic and intuitive expression of the robot's emotional state and social gaze. EDDIE is also equipped with speech processing and synthesis for human-robot interaction. It also contains a pair of Firewire cameras mounted in the eyes, which are used as additional sensors for face and gesture recognition. The other robot is equipped with a pan-tilt stereo camera head.

Apart from the robot on-board sensing, global sensors are used for detection and tracking of robots, dynamic objects, humans and gestures. A cluster of 40 overhead cameras cover the complete experimental area. It enables the tracking of all people and robots as well as the recognition of gestures and objects such as cups, plates and boxes. In addition, part of the area is covered by a Visualeyez^{TM} VZ4000 motion measurement and tracking system by PhoeniX Technologies Inc. [29]. It enables precise tracking of objects and human motion.

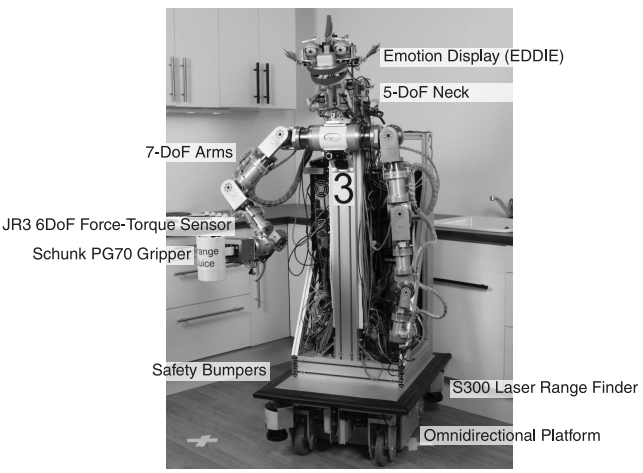


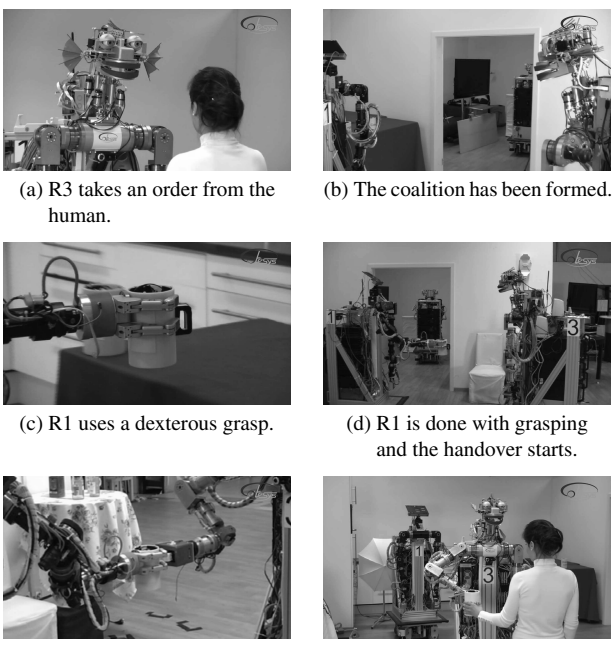
Fig. 7 Robot hardware overview.

5.2 A Cooperative Service Task

The two robots are deployed in a service scenario, where they are supposed to take orders from the present persons and serve cups with the ordered drinks to them. Figure 8 shows six scenes, where the robots successfully complete the task. A full video of the scenario can be viewed on our web-page www.murola.de.

In Fig. 8 (a), R3 has already detected one human, and approaches her to take the order (*ORDER* task). The ordering dialog results in a new task (*SERVE* task) for the multi-robot system. The auctioneer of R3 is aware of two plans to complete the task, one single-robot plan where the subtasks are to grasp the cup and hand it over to the human, and a plan where the two robots have to cooperate. In the two-robot plan, one robot has to grasp the cup and then hand it over to the other robot (*BRING* task) that will deliver it to the human (*FETCH* task). Subsequently, the two plans are announced. Because of the different hardware setups of the robots – R3 has only a two-finger gripper and lower grasp capabilities, while R1 is not equipped with EDDIE and cannot interact with humans – and the position of the cup, the brokers on both robots realize that they cannot complete the single robot plan. However, R1 sends a bid for the two-robot plan, and more specifically for the subtask of grasping the cup, whereas R3 sends one for the subtask of handing it over to the human. After receiving the bids, the auctioneer creates a coalition with the two robots and assigns the tasks, Fig. 8 (b).

R1 has to grasp the drink and hand it over to R3, which



(a) R3 takes an order from the human.

(b) The coalition has been formed.

(c) R1 uses a dexterous grasp.

(d) R1 is done with grasping and the handover starts.

(e) Robot-Robot handover takes place.

(f) The human is served the coke.

Fig. 8 Scenes showing the cooperative serving of a drink by two robots.

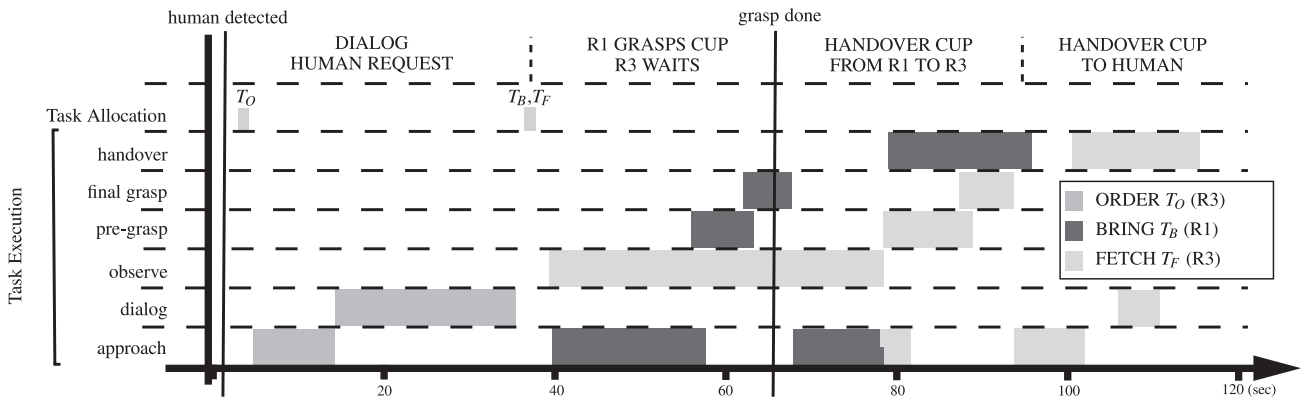


Fig. 9 Timeline of the SERVE-task. The vertical axis represents the primitive tasks. A bar depicts the active time of a primitive task, and bars with same color compose a complex task.

will then serve it to the human. In Fig. 8 (c) R1 uses its dexterous grasping capabilities to grasp the cup. Meanwhile, R3 is waiting for R1 to finish, Fig. 8 (d), and to receive the cup, Fig. 8 (e). Finally, in Fig. 8 (f), R3 gives the human the drink.

Figure 9 presents a detailed timeline of the task. The first label on the vertical axis represents the task allocation algorithm that makes the distributed decision on which robot will execute the pending tasks. The message exchange required by the task allocation takes place using the IceStorm multicast functionality of the *ARCADE* middleware. The other labels on the vertical axis of the figure denote primitive tasks that are used by the task execution to build more complex tasks. For example, the *approach* task is re-used by all the high-level tasks, e.g. *ORDER*, *BRING* and *FETCH*. Additionally, complex tasks consist of a number of simpler tasks that are activated in sequential and parallel manner. During the *BRING* task, the *pre-grasp* subtask that moves the arm to a position to perform the grasp is activated while the *approach* subtask is still active. Collaboration among robots is coordinated by exchanging messages that convey the current state of each robot. R1, uses the distributed communication mechanisms of *ARCADE* to notify R3 that it has successfully completed the grasping of the cup, and that it is ready to hand it over. This is illustrated by the thin vertical line in Fig. 9.

6. Conclusion

The efficiency of robotic systems is essentially determined by their capability of exploiting available information. This requires a middleware for easy data exchange throughout the system as well as tools for efficient and coordinated task execution.

In this article a concise framework is presented, which handles these issues for multi-robot systems. The *ARCADE* middleware enables easy data exchange among all modules throughout the system and supports both real-time local actuator control and efficient communication between robots.

The presented framework has a hybrid centralized-distributed structure, which allows robust and flexible execution of multi-robot tasks. The centralized task execution on each robot provides means to implement complex tasks, and the distributed task allocation yields efficient and cooperative operation of the complete multi-robot system. The applicability of the presented approach is verified in a two-robot service scenario.

Acknowledgements

This work is supported in part within the DFG excellence research cluster *Cognition for Technical Systems - CoTeSys* (www.cotesys.org). Special thanks to the entire MuRoLa team for their support and also to Dr. K. Kühnlenz and S. Sosnowski for the provision of EDDIE.

References

[1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M.N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T.M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.W. Seo, S. Singh, J. Snider, A. Stentz, W.R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," J. Field Robotics, vol.25, no.8, pp.425–466, 2008.

[2] DARPA Grand Challenge, "http://www.darpa.mil/grandchallenge"

[3] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," J. Field Robotics, vol.25, no.9, pp.569–597, 2008.

[4] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An integrated humanoid platform for sensory-motor control," IEEE/RAS International Conference on Humanoid Robots (Humanoids), pp.169–175, 2006.

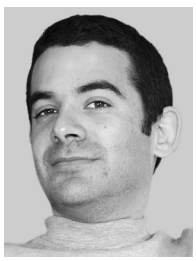
[5] http://www.mca2.org/

[6] K. Okada, T. Ogura, A. Haneda, D. Kousaka, H. Nakai, M. Inaba, and H. Inoue, "Integrated system software for HRP2 humanoid," Proc. 2004 IEEE International Conference on Robotics and Automation, 2004, ICRA '04, vol.4, pp.3207–3212, May 2004.

[7] F. Kanehiro, K. Fujiwara, S. Kajita, K. Yokoi, K. Kaneko, H. Hirukawa, Y. Nakamura, and K. Yamane, "Open architecture humanoid robotics platform," Proc. IEEE International Conference on Robotics and Automation, 2002, ICRA '02, vol.1, pp.24–30, 2002.
[8] T. Matsui, "Multithread object-oriented language EusLisp for parallel and asynchronous programming in robotics," IEEE 6th Symposium on Parallel and Distributed Processing, 1994.
[9] "Object management groups," http://www.omg.org/
[10] N. Kalra, M.B. Dias, R.M. Zlot, and A.T. Stentz, "Market-based multirobot coordination: A comprehensive survey and analysis," Tech. Rep. CMU-RI-TR-05-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Dec. 2005.
[11] B.P. Gerkey and M.J. Matarić, "Sold!: Auction methods for multirobot coordination," IEEE Trans. Robot. Autom., vol.18, no.5, pp.758–768, Oct. 2002.
[12] L.E. Parker and F. Tang, "Building multi-robot coalitions through automated task solution synthesis," Proc. IEEE, vol.94, no.7, pp.1289–1305, 2006.
[13] P. Gmytrasiewicz, , P.J. Gmytrasiewicz, and P. Doshi, "A framework for sequential planning in multi-agent settings," J. Artificial Intelligence Research, vol.24, pp.24–49, 2004.
[14] S. Seuken and S. Zilberstein, "Formal models and algorithms for decentralized decision making under uncertainty," Autonomous Agents and Multi-Agent Systems, vol.17, no.2, pp.190–250, 2008.
[15] P. Stone and M.M. Veloso, "Multiagent systems: A survey from a machine learning perspective," Autonomous Robots, vol.8, no.3, pp.345–383, 2000.
[16] B. Gerkey, R.T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," Proc. 11th International Conference on Advanced Robotics (ICAR), pp.317–323, Coimbra, Portugal, June 2003.
[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T.B. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, "ROS: An open-source robot operating system," International Conference on Robotics and Automation, Open-Source Software Workshop, 2009.
[18] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," Robotics and Autonomous Systems, vol.56, no.1, pp.29–45, 2008.
[19] H. Bruyninckx, "Open robot control software: The OROCOS project," Proc. IEEE International Conference on Robotics and Automation (ICRA), pp.2523– 2528, Seoul, Korea, May 2001.
[20] D. Althoff, O. Kourakos, M. Lawitzky, A. Mörtl, M. Rambow, F. Rohrmüller, D. Brščić, S. Hirche, and M. Buss, "An architecture for real-time control in multi-robot systems," 3rd International Workshop on Human-Centered Robotic Systems, 2009.
[21] M. Henning, "A new approach to object-oriented middleware," IEEE Internet Comput., vol.8, no.1, pp.66–75, 2004.
[22] M. Goebl and G. Färber, "A real-time-capable hard- and software architecture for joint image and knowledge processing in cognitive automobiles," Proc. 2007 IEEE Intelligent Vehicles Symposium, pp.734–740, June 2007.
[23] M. Goebl, Eine realzeitfähige Architektur zur Integration kognitiver Funktionen, Ph.D. Thesis, Technische Universität München, Institute for Real-Time Computer Systems, 2009.
[24] L. Sciavicco and B. Siciliano, Modelling and Control of Robot Manipulators, Springer Verlag, 2000.
[25] F. Rohrmüller, G. Lidoris, D. Wollherr, and M. Buss, "System interdependence analysis for autonomous mobile robots," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.
[26] U.D. Hanebeck, N. Saldic, and G. Schmidt, "A modular wheel system for mobile robot applications," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp.17–23, 1999.
[27] B. Stanczyk, Developement and Control of an Anthropomorphic Telerobotic System, Ph.D. Thesis, Technische Universität München, Institute for Automatic Control Engineering, 2006.
[28] S. Sosnowski, A. Bittermann, K. Kühnlenz, and M. Buss, "Design and evaluation of emotion-display EDDIE," Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06), pp.3113–3118, Beijing, China, Oct. 2006.
[29] PhoeniX Technologies Incorporated, "www.ptiphoenix.com"

Matthias Rambow is currently a research assistant at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. He received the diploma degree in Computer Science in 2008 from the Technische Universität Karlsruhe. His research interests include autonomous mobile robotics and multi-sensor object exploration.

Florian Rohrmüller is currently a research assistant at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. He received the diploma engineer degree in Electrical Engineering in 2007 from the Technische Universität München. His research interests include autonomous mobile robotics and decision making in multi-robot systems under uncertainty.



Omiros Kourakos is currently a research assistant at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. He received the M.Sc. degree in Electrical Engineering in 2007 from the Technische Universität München. His research interests include autonomous robotics and mobile manipulators.

Dražen Brščić received his diploma and master degree in Electrical Engineering in 2000 and 2004 from the University of Zagreb, Croatia, and the doctor degree from The University of Tokyo, Japan in 2008. He is currently a senior researcher at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. His research interests include mobile robotics, intelligent spaces and object tracking.

Dirk Wollherr is senior researcher and lecturer at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. He received his diploma and doctor degree in Electrical Engineering in 2000 and 2005 from TU München. From 2001–2004 he has been research assistant at the Control Systems Group, Technische Universität Berlin, Germany. The Japanese Society for the Promotion of Science (JSPS) granted a research fellowship in 2004 at the Yoshihiko-Nakamura-Lab, The University of Tokyo, Japan. From 2006–2008 Dr. Wollherr has been General Manager of the Cluster of Excellence "Cognition for Technical Systems (CoTeSys)". His research interests include automatic control, robotics, autonomous mobile robots, human-robot-interaction, and humanoid walking.

Sandra Hirche received the diploma engineer degree in Mechanical Engineering and Transport Systems in 2002 from the Technical University Berlin, Germany, and the Doctor of Engineering degree in Electrical Engineering and Computer Science in 2005 from the Technische Universität München, Munich, Germany. From 2005–2007 she has been a JSPS (Japanese Society for the Promotion of Science) PostDoc at the Tokyo Institute of Technology, Tokyo, Japan. Since 2008 she holds an associate professor position at the Institute of Automatic Control Engineering, Technische Universität München. Her research interests include control over communication networks, networked control systems, cooperative control, human-machine interaction, mechatronics, multimodal telepresence systems and perception-oriented control.

Martin Buss is currently full professor at the Institute of Automatic Control Engineering, Technische Universität München, Munich, Germany. He received the diploma degree in 1990 from TU Darmstadt, Germany, and the doctor degree from the University of Tokyo, Japan, in 1994. As a postdoctoral researcher he stayed with the Department of Systems Engineering, Australian National University, Canberra, in 1994/5. From 1995–2000 he has been senior research assistant and lecturer at the Institute of Automatic Control Engineering, TU München, where he finished his habilitation dissertation in 2000. From 2000–2003 he was full professor at TU Berlin, Germany. His research interests include automatic control, robotics, intelligent control, multi-modal human-system interfaces, haptic systems, optimization, nonlinear and hybrid systems.