Team Homer@UniKoblenz — Approaches and Contributions to the RoboCup@Home Competition

Viktor Seib(✉), Stephan Manthe, Raphael Memmesheimer, Florian Polster, and Dietrich Paulus

Active Vision Group (AGAS), University of Koblenz-Landau, Universitätsstr. 1, 56070 Koblenz, Germany
{vseib,smanthe,raphael,fpolster,paulus}@uni-koblenz.de
http://agas.uni-koblenz.de, http://homer.uni-koblenz.de

Abstract. In this paper we present the approaches and contributions of team homer@UniKoblenz that were developed for and applied during the RoboCup@Home competitions. In particular, we highlight the different abstraction layers of our software architecture that allows for rapid application development based on the ROS actionlib. This architectural design enables us to focus on the development of new algorithms and approaches and significantly helped us in winning the RoboCup@Home competition in 2015. We further give an outlook on recently published open-source software for service robots that can be downloaded from our ROS package repository on http://wiki.ros.org/agas-ros-pkg.

Keywords: RoboCup@Home · Service robots · homer@UniKoblenz · Robot lisa · Robotic architecture

1 Introduction

This paper introduces our team homer@UniKoblenz and its scientific approaches and contributions to the RoboCup@Home competition. Its predecessor team, resko@UniKoblenz, was founded in 2006 by the scientific supervisor and the students of a practical course in the curriculum of the Active Vision Group (AGAS). The team and its robot *Robbie* participated in the RoboCup Rescue league. Among others, resko@UniKoblenz won twice the RoboCup Rescue championship in the category "autonomy" (2007 and 2008).

Because of rising demands on hardware engineering in the Rescue league we put our focus on the @Home league in 2008. This league offered a greater opportunity to follow the original research interests of our group, vision and sensor data processing, while having significantly less demands on hardware engineering and mechatronics. While using the rescue robot for the @Home competition in 2008, one year later we built our service robot *Lisa*. In the beginning, Lisa's software was largely based on Robbie's, however, over time many new components

L. Almeida et al. (Eds.): RoboCup 2015, LNAI 9513, pp. 83–94, 2015.
DOI: 10.1007/978-3-319-29339-4_7

were added, specializing on typical at home tasks like human-robot interaction, object perception and manipulation.

Since our first participation in the RoboCup@Home league, Lisa and homer-@UniKoblenz won many awards, among others the Innovation Award (2010) and the Technical Challenge Award (2012). However, the greatest success in the team's history was the 1st place in the RoboCup@Home competition in 2015.

So far, we collaborate with the RoboCup teams Pumas [5] and Golem [9] by organizing research visits and workshops on the Robot Operation System (ROS) [12]. Further, we published some of the integral software components needed to create a service robot on our ROS package repository website[1]. These components are explained in great detail in a contributed chapter in a book on ROS [4].

To this day, the team consists of students pursuing their Bachelor and Master studies and is supervised by a PhD student of the Active Vision Group. Consequently, most of the students leave the team after only one year and new students take their places. Therefore, it is very important to provide an architecture that is easy to learn and to maintain to allow new students to start developing new algorithms very quickly. This paper introduces our approaches to accomplish these goals and is structured as described in the following.

Section 2 shortly introduces our robot Lisa and its hardware components. The software architecture used by our team is presented in Sect. 3. Different abstraction layers are introduced and related to the service robot architecture structure proposed by Pineda et al. [10]. The algorithms and contributions used by our team in the competitions are explained in Sect. 4. This is followed by Sect. 5 that presents recent and ongoing research in the Active Vision Group that is incorporated into our robot's software. Finally, Sect. 6 summarizes this paper and gives an outlook to our future plans and research.

2 Service Robot Lisa

In this year's competition we used two robots (Fig. 1). The blue Lisa is our main robot and is built upon a CU-2WD-Center robotics platform[2]. The old Lisa serves as an auxiliary robot and uses the Pioneer3-AT platform. Every robot is equipped with a single notebook that is responsible for all computations. Currently, we are using a Lenovo Thinkpad W520 equipped with an Intel Core i7-2670QM processor, 12 GB of RAM with Ubuntu Linux 14.04 and ROS Indigo.

Each robot is equipped with a laser range finder (LRF) for navigation and mapping. A second LRF at a lower height serves for small obstacle detection.

The most important sensors of the blue Lisa are set up on top of a pan-tilt unit. Thus, they can be rotated to search the environment or take a better view of a specific position of interest. Apart from a RGB-D camera (Asus Xtion) and a high resolution RGB camera (IDS UI-5580CP-C-HQ), a directional microphone (Rode VideoMic Pro) is mounted on the pan-tilt unit.

[1] AGAS ROS packages: http://wiki.ros.org/agas-ros-pkg.
[2] Manufacturer of our robotic platform: http://www.ulrichc.de.

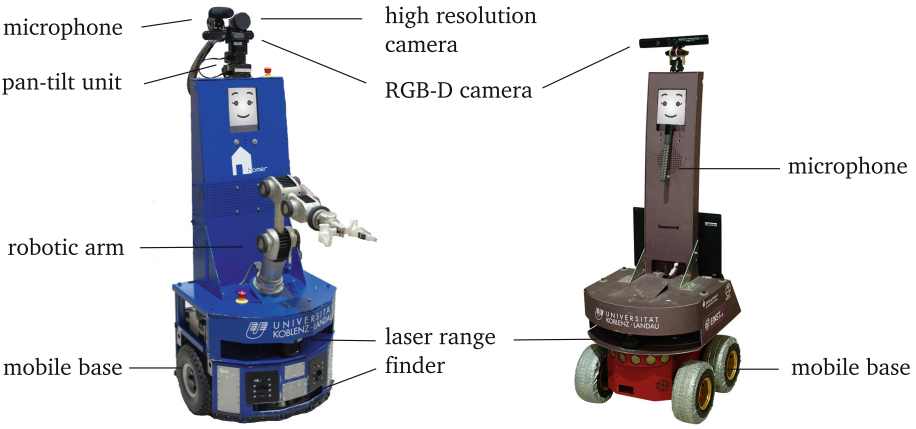

Fig. 1. Lisa (in blue) on the left is our main robot. The Lisa (right) serves as auxiliary robot (Color figure online).

A 6 DOF robotic arm (Neuronics Katana 400HD) is used for mobile manipulation. It is certified for a safe operation around humans and is able to manipulate light-weight objects up to 0.5 kg. The end effector is a custom setup and consists of 4 Festo Finray-fingers.

Finally, a Raspberry Pi inside the casing of the blue Lisa is equipped with a 433 MHz radio emitter. It is used to switch device sockets and thus allows to use the robot as a mobile interface for smart home devices.

3 Software Architecture

Since the foundation of our team we were using a message-based software architecture. In the beginning we used the middleware *Robbie* [22] that was developed in our group. Robbie is a modular architecture that uses messages to exchange data between modules. Each module runs in its own thread and interfaces libraries, hardware drivers or third-party applications. With the advent of ROS we started a step by step conversion of our software modules, which was finished in 2014. Because of the similar design ideas in both architectures, the conversion was easy to manage. Using ROS as middleware facilitates the integration of existing packages, but also sharing our own code with the growing ROS community. Further, we do not need to maintain the middleware and, thus, are able to concentrate on developing new algorithms. This is particularly important, since each year many experienced students are replaced by new ones. However, like in the time of the Robbie architecture, we still adhere to the idea of encapsulating algorithms in stand-alone libraries that are connected by ROS nodes to the application.

In the following we want to highlight some aspects of our architecture and introduce our approaches in implementing RoboCup@Home tests.

3.1 Abstraction Levels

Although our software mainly consists of ROS packages and libraries, several different abstraction layers can be distinguished. We identify the following levels: the functionality level, the behavior level and the application level. Components at each abstraction level are allowed to use only modules from the same or a lower level.

The lowest level is the functionality level. It is formed by nodes that directly access a library or a hardware driver to fulfill one functionality or execute a specific algorithm. Examples are interfacing the robot platform or a sensor, but also executing algorithms like path planning or object recognition. To avoid fragmentation into too small components, these nodes are not limited to exactly interface one algorithm or device. For instance, the node responsible for planning an arm trajectory for grasping also accesses the controller that is responsible to execute the planned path. All these nodes have a rather simple structure, mostly taking a request and providing a response. Because of this type of utilization in most cases it is favorable to use ROS services as interfaces, rather than messages.

The second level is the behavior level. This level is composed by nodes employing the ROS *actionlib*[3]. The actionlib provides standardized interfaces to define and execute (long running) preemptable tasks. While the task is executed, periodic feedback is provided on the state of the task execution. In our system, actions are defined at different levels of granularity. Some actions merely encapsulate one functionality (e.g. voice synthesis), while others contain more complex tasks like grasping an object. The latter example includes detecting and recognizing the object, positioning the robot accordingly and finally moving the arm while avoiding obstacles. System components on this abstraction level are allowed to use other actions (mostly of minor granularity) to facilitate composition of more complex behaviors. In general, complex actions are implemented using state machines that react to different outcomes of sub-task execution. We are currently working on integrating more versatile error handling and feedback for every action.

The third and highest abstraction level is the application level. This level contains predefined or dynamic sequences of behaviors and functionalities, e.g. the RoboCup@Home tests. At this level, a high-level task specification language as proposed in [11] or [24] might be handy to facilitate complex behavior description and application composition. In our case we decided to use the same programming language (C++) as was used in all other abstraction levels, to facilitate the integration of new students. However, we implemented special encapsulating interfaces to be used at this level that significantly facilitate application implementation as compared to pure state machines and ROS interfaces. Subsection 3.3 provides more details on these interfaces.

[3] ROS actionlib documentation: http://wiki.ros.org/actionlib.

3.2 Relation to Service Robot Concepts

In [10] Pineda et al. discuss a concept and functional structure of a service robot. They identify the following three system levels in service robot design: system programming, robotic algorithms and the functional specification. These three system levels are related to Newel's levels of computational systems [8], stating that the first two layers can be reduced to more basic components. However, the functional specification level emerges from the lower levels providing additional knowledge about a task structure and, thus, can not be reduced to components of lower levels. For this reason, Pineda et al. propose a specialized task specification language [11].

They further propose two functionality layers: *Application* and *Behaviors*. The Application layer, being the higher layer, defines the task structure and conceptually corresponds to our application level. The Behaviors layer contains structured hierarchical objects corresponding to basic tasks and conceptually resembles our two basic levels: the functionality and the behavior level. In the hierarchical objects on the Behavior layer of Pineda et al., the lowest hierarchy level can be compared to our functional level, whereas the higher hierarchies are similar to our behavior level. However, the exact relation of our system levels to the ones proposed by Pineda et al. is not that clear in this case.

Another crucial component in service robot design identified in [10] is task management. Task management allows the robot to assess its current state, react to unexpected situations and perform proper error handling and recovery. To this date our system architecture lacks a unified task management and task planning component. Although basic task management strategies exist, they are loosely coupled and do not properly relate to the abstraction layers of our system architecture. Since task management is an important component, we are currently focusing on developing a task management concept that best suits our demands.

3.3 Implementing RoboCup@Home Tests

RoboCup@Home tests consist of different behaviors and functionalities that are combined in a specific way to create a desired application. Hence, they reside on the application abstraction level. Pineda et al. [10] identify two different types of RoboCup@Home tests: static and dynamic tasks. While almost all RoboCup@Home tests describe a predefined sequence of actions and are therefore considered static, the *General Purpose Service Robot* (GPSR) test is the only one with a dynamic task structure.

To enable simple static task implementation, we have created a utility module that encapsulates all behaviors our robot can execute. The most simple test is the *Robot Inspection*, where the robot is waiting in front of a closed door. As soon as the door is opened, the robot navigates to a predefined inspection position. After being inspected the robot receives a speech command to exit the arena. In our software architecture, the implementation for this test would look like this:

```
ros::NodeHandle n;

Robot lisa(n);
lisa.waitForDoor();
lisa.navigateTo("inspection");
lisa.speak("I am ready for inspection");
lisa.speak("Say 'exit now' if you want me to exit");
lisa.waitForCommand("exit now");
lisa.navigateTo("exit");
```

Of course this is a simple example where the robot does not have to react to the result of one command to decide which command is next or which parameters to pass to the next command. In practice, even static tasks are rich in control structures like loops and conditional clauses. However, these programs are still easy to read and task oriented, by hiding the implementation details.

Note that all of the behaviors in the above example are blocking. The next command is only executed when the previous one was completed. Nevertheless, sometimes a non-blocking version is required, which is also available in the provided interface. The downside of this approach is that this convenience module gets very complex and requires careful maintenance. We are currently considering several ideas on how to improve these interfaces.

Unlike the other tests, the GPSR test does not have a predefined task sequence. This test is divided into three types of complexity. The first type is an unknown sequence of commands from a known set that need to be executed in the correct order. The second type is similar to the first one, but might contain missing information that the robot must infer on its own or optionally ask for clarification. The third and most complex type deals with incomplete and erroneous information that requires task planning and complex situation analysis.

To solve this test, we use a natural language parser with keyword spotting. Individual tasks and the corresponding parameters are extracted from the given command and executed one after another. Missing information is obtained from a knowledge base containing a hierarchical definition of objects, categories and places. So far all of these information is inserted manually into the knowledge base after being specified by the organizers of the competition. However, this simple approach only allows for solving the first and second type of GPSR commands. The third type requires more complex reasoning and task management which is subject of ongoing work.

4 Approaches and Contribution

In this section we briefly describe the software that we recently contributed to the RoboCup and ROS community. More details are provided in our current team description paper [18] and a detailed tutorial in [4]. Further, we describe

other algorithms and approaches that we use to realize the functionalities and behaviors needed for typical @Home tasks.

4.1 Software Contributions

We offer ROS packages for the following tasks: Mapping and Navigation [25] based on [27] (Fig. 2), Object Recognition [15] (Fig. 4), Speech Recognition and Synthesis and a Robot Face [13] (Fig. 3) for human robot interaction. Additionally, a Graphical User Interface provides convenient interfaces to access the provided functionalities. Video tutorials are available online[4].

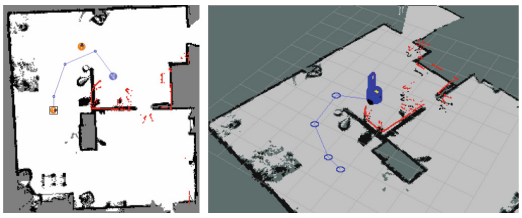

Fig. 2. Mapping and navigation example

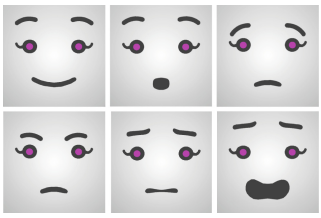

Fig. 3. Different face expressions

4.2 Approaches and Algorithms

In this Section we introduce our approaches to some important areas of functionalities.

Speech Recognition. For speech recognition we integrated and evaluated different approaches. In the past we used *PocketSphinx*[5] for speech recognition. To improve recognition results, different dictionaries were activated depending on the current state of the task (e.g. if the robot was asking for a drink order only drink names could be understood).

Another approach we tried was the integration of the Android speech recognition. Although an internet connection improves recognition results, it is not mandatory. The main benefit of this approach is that no dictionary needs to be defined and the robot will be enabled to understand everything. This opens different scenarios towards general purpose robotics.

Finally, the approach we are currently using is the grammar based VoCon speech recognition by Nuance[6]. We use the begin and end of speech detection

[4] homer@UniKoblenz channel with video tutorials: https://www.youtube.com/user/homerUniKoblenz.

[5] Speech recognition system PocketSphinx: http://www.speech.cs.cmu.edu/pocketsphinx/.

[6] VoCon speech recognition: http://www.nuance.com/for-business/speech-recognition-solutions/vocon-hybrid/index.htm.

Fig. 4. Object recognition example

to improve recognition results and discard results below a certain confidence threshold. With the Nuance VoCon approach we were able to win the speech recognition benchmark task at the RoboCup 2015 in Hefei.

Multi-robot Coordination. Our first approach to multi-robot coordination aimed at a hierarchical structure, where one robot being the master sent commands to the slave robot [14]. The good speech recognition results we achieved this year allowed for a new experiment. To our best knowledge, we showed the first robot-robot interaction in the RoboCup@Home using natural language. In the final demonstration both robots assigned tasks to one another using synthesized speech and recognizing it. We used a WiFi fallback in case a command was not understood in a certain time. As the human voice is already a general interface for service robots in RoboCup@Home, we showed that it can be used for robots as well. Furthermore, it is imaginable to use it as a general robot-robot interaction interface, even for robots of different teams.

Of course, data that can not be expressed in human language (e.g. object models, slam maps) has to be exchanged in a traditional way. However, if robots communicated using natural language e.g. for task assignment, people would be aware of what the robot is doing and could even provide modifications to previously assigned tasks ("Set the table for one more person, we will have a guest.").

People Detection and Tracking. People are detected by the combination of three sensors. The laser range finder is used to detect legs, while the RGB camera image provides data for face detection. We use the face detection algorithm implemented in the OpenCV library[7]. Finally, the depth camera allows to detect silhouettes of persons [20].

For people tracking we use rich RGB-D data from a depth camera. The sensor is mounted on a pan-tilt unit to actively follow the current person of interest.

[7] OpenCV face detection: http://docs.opencv.org/2.4/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html.

Our people tracker is based on the publicly available 3D people detector of Murano et al. [7] in combination with online appearance learning using adaboost classifiers on color histograms. We estimate the target position and velocity using a linear Kalman filter with a constant velocity motion model. At every time step, we select the detection with highest classification score inside the gating region for target association and update the classifier with positive and negative samples from the set of current detections accordingly. Occlusion detection is based on classification scores as well. We perform Kalman update and appearance learning only if the highest classification score exceeds a given threshold.

Object Detection and Recognition. Objects for mobile manipulation are detected by first segmenting horizontal planes as table hypotheses from the point cloud of the RGB-D camera. Subsequently, all points above the plane are clustered and the resulting clusters considered as objects. We combine the advantages of the point clouds from the RGB-D camera with the high resolution images from the IDS camera by calculating the correspondences between the points in the point cloud and the pixels in the color image. To accomplish this, the relative pose between the depth sensor and the color sensor of the camera was calculated with a stereo calibration algorithm [2]. With the knowledge about the relative poses between the two sensors we were able to project the position of detected objects in the point cloud into the color images and use a high resolution image patch from the IDS camera for object recognition.

Transparent objects (in our case drinking glasses) are detected by making use of one fault of the structured light sensor. The light emitted by the RGB-D camera is scattered in transparent objects providing no valid depth information. We segment areas with no depth data and compare them with drinking glass contour templates. Since the supporting table plane around the transparent objects has valid depth information, a size and location estimation of the transparent objects is obtained and used for grasping. However, we are currently conducting experiments with a deep convolutional neural network to augment recognition and detection results of transparent objects.

Object Manipulation. We use an A* search algorithm within the arm configuration space to plan an arm trajectory for grasping. To avoid collisions, we run an on-the-fly collision detection in which we check a simplified arm model represented by a set of line segments against a kd-tree build on the depth image of the environment. The planned path is smoothed with a recursive line fitting algorithm in the configuration space. The resulting path consists of a minimal set of configurations which are directly connected by linear moves in the configuration space. Although our approach works well, we are not satisfied with the runtime of the planning algorithms, especially if lots of obstacles are present. Therefore, we are integrating the MoveIt library[8] which promises a flexible framework with many different algorithms to choose from.

[8] MoveIt library: http://moveit.ros.org/.

5 Current Research

Apart from the various fields mentioned in the previous Sections that we are currently working on the main focus is put into computer vision. The two most important topics are highlighted in this section.

Fig. 5. Sitting affordance detection with suggested human body pose.

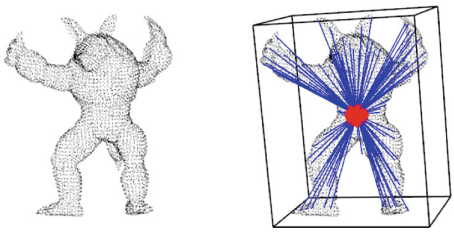

Fig. 6. Recognition result and estimated bounding box.

3D Object Recognition. Leibe et al. presented the Implicit Shape Model (ISM) algorithm as an extension to the popular bag-of-keypoints approach in [6]. Recently, extensions of the ISM approach to 3D data have been proposed [3,26]. Unlike in related work on 3D ISM, we use a continuous Hough-space voting scheme. In our approach [16] (Fig. 6), SHOT features [23] from segmented objects are learned. However, contrary to the ISM formulation, we do not cluster the features. Instead, to generalize from learned shape descriptors, we match each detected feature with the k nearest learned features in the detection step. Each matched feature casts a vote into a continuous Hough-space. Maxima for object hypotheses are detected with the Mean-Shift Mode Estimation algorithm [1]. In a recent work we compared our algorithm with other approaches [17] and achieved best results. Still, the long runtime of our approach is a drawback that we are currently working on. Apart from this purely shape based approach we are also working on combining 2D and 3D features to improve object recognition results based on RGB-D point clouds [19].

Affordance Detection. Affordances have gained much popularity for object classification and scene analysis. Our current research focuses on sitting affordances to analyze scenes regarding sitting possibilities for an anthropomorphic agent. Recently, we introduced the concept of fine-grained affordances [21]. It allows to distinguish affordances on a fine-grained scale (e.g. sitting without backrest, sitting with backrest, sitting with armrests) and thus facilitates the object classification process. Additionally, our approach estimates the sitting pose with regard to the detected object (Fig. 5).

Our current work on that field focuses on adding more affordances, but also experimenting with other agent models. We plan to extend our method to detecting different fine-grained grasping affordances and thus allow for precise grasp planning for mobile manipulation.

6 Summary and Outlook

In this paper we have presented the approaches developed and applied by our team homer@UniKoblenz with robot Lisa that participates in the RoboCup@-Home competition. A special emphasis was put on our software architecture and the therein subsumed abstraction layers that significantly facilitate application development even for new team members without a robotics background.

We further gave a short introduction into our contributed software packages available at our ROS package repository. As we continue developing and improving our software, more packages will be made available to everyone.

Our plans for the future are twofold. On one hand we will continue research in the computer vision field, focusing on 3D object recognition, affordance detection and transparent object detection. On the other hand we will investigate how a meaningful task management can be included into our software architecture. This is an inevitable step towards general purpose service robots which need to be able to handle and recover from errors, be aware of their environment and react to unexpected situations.

References

1. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
2. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, New York (2003)
3. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3D SURF for robust three dimensional classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 589–602. Springer, Heidelberg (2010)
4. Koubaa, A. (ed.): Robot Operating System (ROS) - The Complete Reference. Studies in Computational Intelligence, vol. 625. Springer, Switzerland (2016)
5. Bio-Robotics Laboratory. Pumas@home 2015 team description paper (2015). http://biorobotics.fi-p.unam.mx/downloads/finish/3-papers/532-tdp2015
6. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: ECCV 2004 Workshop on Statistical Learning in Computer Vision, pp. 17–32 (2004)
7. Basso, F., Munaro, M., Menegatti, E.: Tracking people within groups with RGB-D data. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS) (2012)
8. Newell, A.: The knowledge level: presidential address. AI Mag. **2**(2), 1 (1981)
9. Pineda, L.A., Rascon, C., Fuentes, G., Estrada, V., Rodriguez, A., Meza, I., Ortega, H., Reyes, M., Pena, M., Duran, J., et al.: The golem team, robocup@ home 2014 (2014). http://turing.iimas.unam.mx/golem/pubs/golem-TDP.pdf
10. Pineda, L.A., Rodríguez, A., Fuentes, G., Rascon, C., Meza, I.V.: Concept and functional structure of a service robot. Int. J. Adv. Rob. Syst. **12**, 6 (2015)
11. Pineda, L.A., Salinas, L., Meza, I., Rascon, C., Fuentes, G.: Sitlog: a programming language for service robot tasks. Int. J. Adv. Rob. Syst. **10**, 538 (2013)

12. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
13. Seib, V., Giesen, J., Grüntjens, D., Paulus, D.: Enhancing human-robot interaction by a robot face with facial expressions and synchronized lip movements. In: Skala, V. (ed.) 21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (2013)
14. Seib, V., Gossow, D., Vetter, S., Paulus, D.: Hierarchical Multi-robot Coordination. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS, vol. 6556, pp. 314–323. Springer, Heidelberg (2010)
15. Seib, V., Kusenbach, M., Thierfelder, S., Paulus, D.: Object recognition using hough-transform clustering of surf features. In: Workshops on Electronical and Computer Engineering Subfields, pp. 169–176. Scientific Cooperations Publications (2014)
16. Seib, V., Link, N., Paulus, D.: Implicit shape models for 3d shape classification with a continuous voting space. In: Braz, J., Battiato, S., Imai, F.H. (eds.) VISAPP 2015 - Proceedings of International Conference on Computer Vision Theory and Applications, vol. 2, pp. 33–43. SciTePress (2015)
17. Seib, V., Link, N., Paulus, D.: Pose estimation and shape retrieval with hough voting in a continuousvoting space. In: Gall, J., Gehler, P., Leibe, B. (eds.) GCPR 2015. LNCS, vol. 9358, pp. 458–469. Springer, Heidelberg (2015)
18. Seib, V., Manthe, S., Holzmann, J., Memmesheimer, R., Peters, A., Bonse, M., Polster, F., Rezvan, B., Riewe, K., Roosen, M., Shah, U., Yigi, T., Barthen, A., Knauf, M., Paulus, D.: Robocup 2015 - homer@unikoblenz (Germany). Technical report, University of Koblenz-Landau (2015)
19. Seib, V., Memmesheimer, R., Paulus, D.: Ensemble classifier for joint object instance and category recognition on RGB-D data. In: 2015 22nd IEEE International Conference on Image Processing (ICIP) (2015)
20. Seib, V., Schmidt, G., Kusenbach, M., Paulus, D.: Fourier features for person detection in depth data. In: Azzopardi, G., Petkov, N., Yamagiwa, S. (eds.) CAIP 2015. LNCS, vol. 9256, pp. 824–836. Springer, Heidelberg (2015)
21. Seib, V., Wojke, N., Knauf, M., Paulus, D.: Detecting fine-grained affordances with an anthropomorphic agent model. In: Agapito, L., Bronstein, M.M., Rother, C. (eds.) ECCV 2014 Workshops. LNCS, vol. 8926, pp. 413–419. Springer, Heidelberg (2015)
22. Thierfelder, S., Seib, V., Lang, D., Häselich, M., Pellenz, J., Paulus, D.: Robbie: a message-based robot architecture for autonomous mobile systems. In: INFORMATIK -Informatik Schafft Communities (2011)
23. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 356–369. Springer, Heidelberg (2010)
24. Tousignant, S., Van Wyk, E., Gini, M.: Xrobots: a flexible language for programming mobile robots based on hierarchical state machines. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 1773–1778. IEEE (2012)
25. Wirth, S., Pellenz, J.: Exploration transform: a stable exploring algorithm for robots in rescue environments. In: Workshop on Safety, Security, and Rescue Robotics, vol. 9, pp. 1–5 (2007)
26. Wittrowski, J., Ziegler, L., Swadzba, A.: 3d implicit shape models using ray based hough voting for furniture recognition. In: 2013 International Conference on 3DTV-Conference, pp. 366–373. IEEE (2013)
27. Zelinsky, A.: Environment exploration and path planning algorithms for a mobile robot using sonar. Ph.D. thesis, Wollongong University, Australia (1991)