Lessons Learnt from Developing the Embodied AI Platform CAESAR for Domestic Service Robotics

Alexander Ferrein

Electrical Engineering Department
FH Aachen, Aachen, Germany
ferrein@fh-aachen.de

Tim Niemueller, Stefan Schiffer
and **Gerhard Lakemeyer**
Knowledge-based Systems Group
RWTH Aachen University, Aachen, Germany
{niemueller,schiffer,gerhard}@cs.rwth-aachen.de

Abstract

In this paper we outline the development of CAESAR, a domestic service robot with which we participated in the robot competition RoboCup@Home for many years. We sketch the system components, in particular the parts relevant to the high-level reasoning system, that make CAESAR an intelligent robot. We report on the development and discuss the lessons we learnt over the years designing, developing and maintaining an intelligent service robot. From our perspective of having participated in RoboCup@Home for a long time, we answer the core questions of the workshop about platforms, challenges and the evaluation of integrative research.

Introduction

The key questions in the area of designing intelligent robots raised at the 2013 AAAI Spring Symposium on *Designing Intelligent Robots: Reintegrating AI II* as stated on the Symposium's website are:

Q1: How do we evaluate integrative research?

Q2: What are good challenges and benchmark problems?

Q3: What are good platforms and frameworks for this work?

In this paper, we give an overview of our work on the domestic service robot CAESAR, which successfully participated in RoboCup@Home competitions over several years; we will give answers to the central questions of the Symposium from our perspective of designing, running, and maintaining such a robot for over more than seven years. We will report on the lessons we learnt (w.r.t. these questions) and give an outlook on possible future developments.

The basic hardware platform was developed more than ten years ago with the initial goal of participating in RoboCup soccer competitions, which we did until 2006. However, we did not want to build a platform that is totally specific to soccer. The software as well as the hardware design we had chosen was kind of conservative. It was what we knew about designing an autonomous robot system along the lines of service robots known in the Cognitive Robotics scene. We designed a well-powered differentially driven robot with focus on the software development and high-level reasoning. Facing complications with the rule changes over the years (no walls around the playing field to make localization harder, faster and more light-weight opponents), we designed multi-robot coordination approaches and applied sensor fusion to compensate for our disadvantages. What was unique to our going about in robotic soccer was our approach to apply (and extend) the situation calculus-based language GOLOG for the high-level control of our soccer robots (Ferrein and Lakemeyer 2008). We used this language also to try and formalise (to some extent) robotic (and human) soccer (Dylla et al. 2008).

In 2006, the RoboCup@Home competitions were started with the goal to foster the development of socially assistive robots for domestic environments. Part of the idea of this RoboCup league is to award teams to show ready-to-run solutions which are closer to some product stage (i.e. less duct tape on the robot). With all our software development in place, we then decided to join this effort; in 2006 our robot design was better suited for domestic service robots than for soccer competitions and we made the transition to this new domain (Schiffer, Ferrein, and Lakemeyer 2006). To meet the demands of this new application area we had to extend our hard- and software quite a bit. In this process we touched a broad range of problems; with many of them not being in our research focus.

In the rest of the paper we will give a rough overview of CAESAR. Many details have already been published elsewhere and we give pointers for the interested reader to get some more detailed information. In the next section, we will briefly review the domestic service robot domain and RoboCup@Home competitions. Then we overview CAESAR's hardware design and its software components also w.r.t. crucial design decisions that we took, before we summarise the lessons we learnt during this long period of developing, maintaining, and re-designing the system. We conclude with a summary of our answers to the introductory questions and an outlook on what, in our eyes, is important for a future (version of our) system.

The Domestic Service Robot Domain and the RoboCup@Home Competitions

The evolution of our robot platform CAESAR over the years is shown in Fig. 1. It is iteratively re-designed and re-built to operate in human-populated environments, where it should

(a) 2006 (b) 2007 (c) 2009 (d) 2011

Figure 1: The Evolution of CAESAR

Figure 2: CAESAR performing the *WhoIsWho* test looking for people in the RoboCup@Home arena (2008).

be helpful around the house, ultimately assisting elderly or disabled people with their daily activities.

Apart from the requirements commonly set for an autonomous mobile robot in general, its application in domestic service robotics (DSR) scenarios with humans around places additional demands. Two important issues in this regard are flexible and robust intelligent behaviour and human-robot interaction (HRI). When a robot and humans share the same habitat and are working together, the need for HRI is a given and the robot must exhibit some form of intelligent behaviour to be helpful and to work for extended periods of time. At the same time it needs to be robust against various types of variations and errors in its environment, not only those caused by the humans. As an example for the former kind of variations consider that humans are messy (from a robot's perspective) and tend to leave things in different places. They move around items frequently so that the environment is not as predictable as, say, with industrial settings. For the latter kind of variations and errors, recall that a robotic system for complex tasks is a complex system itself. Modules might crash and components might get stuck in certain situations. Robustness against those problems allows for enduring autonomy which is crucial when the robot needs to assist a person over extended periods of time. A domestic service robot has to meet the cognitive demands that arise in daily chores, where complex problems sometimes require deliberation. Strictly pre-defined behaviours are prone to errors and tend to fail since they cannot account for more or less subtle variations and the uncertainty inherent in real-world scenarios all the time. From a human-robot interaction perspective, robots need to be operable by laymen and they need to account for imprecision and fallibility of humans, in general, and of elderly people, in particular.

RoboCup@Home. There are various efforts for benchmarking in domestic service robotics. One of them is the RoboCup@Home competition which particularly focuses on the technologies and techniques for domestic service robots. In annual competitions, researchers from all over the world gather to showcase their latest developments in a number of preset challenges, which become more and more complex from year to year. Since 2006, the RoboCup@home initiative fosters research in artificial intelligence, robotics, and related fields under the roof of RoboCup. It specifically targets the application of autonomous mobile robots as assistive technology in socially relevant tasks in home environments. It is designed to be both, a scientific competition and a benchmark for domestic service robots (Wisspeintner et al. 2009). The general idea in the @Home competition is to let robots perform a set of tasks in home-like scenarios that are as realistic as possible. An example of such a scenario is depicted in Fig. 2, which shows a scene where the robot is driving around in the apartment with the goal to find and to recognise people. The tasks are oriented towards real-life applications of domestic service robotics.

The tests in DSR, in general, and in RoboCup@Home, in particular, require working solutions to specific (sub)tasks such as localization, navigation, face recognition and others. What is even more important, a successful system has to fully integrate those capabilities to a complete system that also has means of deliberation to purposefully work towards achieving a particular mission in a home-like scenario. Our team participated quite successfully in these competitions since they were established. We were able to win the world championship in 2006 and 2007, and became second in 2008. We also won the RoboCup German Open competition in the @Home league in 2007 and 2008.

The Robot CAESAR

Our robot CAESAR is based on a platform initially designed to compete in the RoboCup soccer competitions. Over the years it evolved to a capable robot for domestic service robotics. The changes over the years are depicted in Fig. 1. The base platform has a size of $40\ \text{cm} \times 40\ \text{cm} \times 60\ \text{cm}$, CAESAR's total height is $180\ \text{cm}$. Its main sensor for navigation and localization is a laser range finder. In its current state it features an RGB-D camera on a pan-tilt unit as a second main sensor for perception. A 5 DOF anthropomorphic

arm allows for manipulating things in the world.

Some Features of our Intelligent Robot

To call a robot intelligent, it needs to be equipped with a number of software modules to exhibit somewhat flexible and intelligent behaviour. Among the basic tasks an intelligent robot in a domestic environment (and beyond) has to accomplish is to localize itself in the apartment, to navigate safely around the furniture and to avoid bumping into people. However, this does not yet make a robot appear intelligent. On top of that it needs to be able to communicate in an as natural way as possible with the humans around it and it also needs to take intelligent decisions.

The AI methods used on the different levels of our system which make for a successful domestic robot are:

- base components: localization, collision avoidance, path planning, manipulation
- human-machine interaction: face detection, gesture detection, natural language processing, sound localization
- high-level control: (qualitative) reasoning with the robot programming and plan language READYLOG interfacing the base components through the Behavior Engine.

Base Components

We start our report with the base components and with a description of the robot middleware Fawkes which is the glue between all these components.

Robot Software Framework Fawkes. We are using Fawkes (Niemueller et al. 2010) for most of our components. Fawkes is an Open Source software framework providing a communication middleware, libraries, and components for typical robot applications.[1] It follows the component-based software design paradigm. Each functional entity in Fawkes is a component, implemented as a plugin (consisting of one or more threads) that can be loaded at run-time. Threads have aspects to access functionality (like configuration or logging) asserted and maintained by the framework. Information is exchanged via a hybrid blackboard with transaction semantics for accessing data structures. It can also be accessed over the network and features data introspection. Fawkes comes with a number of useful plugins for tasks like performance evaluation, platform support, and functional components for perception and actuation.

Collision Avoidance. A mobile robot, especially when operating in human-populated environments, needs to avoid collisions with furniture or humans and it has to be safe in this regard. On CAESAR, we deployed a method for local navigation and collision avoidance that is safe and reactive. A local map is derived from the laser range finder's readings. In this map, a collision-free path to the given target point is sought for by making use of an A* search approach. A* search is then also used to find a realisation of this path by means of feasible motion trajectories. The collision avoidance module runs with a frequency of 20 Hz. It recomputes the path and the motion commands every cycle. We deployed this approach for many years in robotics competitions both in robotic soccer as well as in domestic service robotics settings. It was a key to succeed in the domestic robot competition RoboCup@Home several years.

Localization. An intelligent robot has to know where it is located in its environment. Our self-localization implements a Monte Carlo Localization following (Thrun et al. 2001). It approximates the position estimation by a set of weighted samples. Each sample represents one hypothesis for the pose of the robot. To be able to localize robustly based on laser data, we modified the Monte Carlo approach. To allow for the integration of a whole sweep from the LRF, we use a heuristic perception model. With this we are able to localize with high accuracy both in the ROBOCUP environment as well as in home and larger scale office environments.

Path planning. On top of the metrical map used for localization, we additionally keep a topological map that stores information about the positions of and the connections between semantic places like a room or a door. To plan a path from one place to another we perform an A*-search on the topological graph. This yields a list of way-points along which the robot is to navigate to the target place. The local navigation between those way-points is then performed by our collision avoidance method described above.

Mobile Manipulation. For physical interaction with the world CAESAR cannot only move around, but it can also manipulate objects with its robotic arm. To do so, it perceives its surrounding with an RGB-D camera. From the depth information a local model of the scene is generated that is used in the motion planning for the arm. We use OpenRAVE (Diankov and Kuffner 2008) to plan collision-free motion trajectories for the manipulator to pickup objects and to place them somewhere else. To plan such trajectories, e.g. for picking up a cup from a table with multiple objects on it, takes in the order of a few seconds.

Human-Machine Interaction

Apart from the base components mentioned so far, we also needed to add components to increase the interaction capabilities of CAESAR.

Face Detection, Recognition, and Learning. We employ an AdaBoost-based method for face detection (Viola and Jones 2001; Lienhart and Maydt 2002) which is readily available in OpenCV. Further, we have developed an integrated approach for face detection and recognition using random forests (Breiman 2001) where face recognition can also be used separately. The recognition framework is able to integrate new identities to its database on the fly. We use Haar features similar to those used in the boosted face detection cascade by Viola and Jones (Viola and Jones 2001). They allow for fast evaluation and they are known to be good features for face detection and recognition. We grow random trees for recognition iteratively by selecting one test from a set of L candidates that were randomly generated. The best test is chosen by maximising the entropy gain (similarly as in decision tree learning) and the left and right branches are appended to the existing node; the procedure continues until

[1] Find Fawkes at `http://www.fawkesrobotics.org/`

the leaf nodes maintain training data of only a single class. The training is very fast and the recognition performance of the resulting random forests is usually sufficient for our target scenarios. For example, it takes only about 7.5 ms to create a random forest consisting of ten random trees, each grown up to a 1000 nodes on a training collection of 464 face images and six identities to yield a recognition accuracy of over 85 %.

Speech Recognition. Spoken commands can be given to our robot CAESAR conveniently using natural language. We use the SPHINX speech recognition software system from Carnegie Mellon University (CMU). It is based on Hidden Markov Models (HMM). An overview of an early version of SPHINX is given in (Huang et al. 1993). We have built a robust speech recognition system on top by first segmenting closed utterances potentially directed to the robot. These are decoded with two different decoders which run in parallel. One decoder uses a finite state grammar (FSG), the other one is a TriGram decoder. Applying both in parallel, we seek to eliminate each decoder's drawbacks retaining its benefits. We can look for similarities between the FSG's output and the N-best list of the TriGram decoder. This allows for highly accurate recognition in domains which can be captured with a limited vocabulary and grammar. At the same time, false positives, which are immanent in the noisy environments one is confronted with at ROBOCUP competitions, can be filtered out reliably.

Sound Localization. As a valuable input cue in HRI we use a biologically inspired sound source localization. In order to obtain reliable directional information two microphones are used. Although the task would be easier with more microphones, we deliberately chose to restrict ourselves to two because the processing of only two signals is computationally less expensive and standard, off-the-shelf hardware can be used. Furthermore, two microphones are easier to fit on a mobile robotic platform than a larger array. We investigated the combination of our existing sound localization system with the robot's knowledge about its environment, especially the knowledge about dynamic objects. By combining several sensor modalities, sound sources can be matched to objects, thus enhancing the accuracy and reliability of sound localization. (Calmes et al. 2007)

Gesture Recognition. Speech recognition provides an intuitive means to control and interact with a robot. However, a huge part of meaning in communication is also transferred via non-verbal signals. An important mode of this non-verbal communication are gestures, in particular in interaction with a domestic service robot, since controlling the robot often relates to entities in the world such as objects and places or directions. References to objects can conveniently be made by pointing gestures while other, dynamic gestures can be used to indicate directions or commands. In CAESAR's gesture recognition system, we implement a modular architecture where the overall process is decomposed into sub-tasks orchestrated in a multi-step system. We start with a hand detection followed by a hand posture classification. Then we track the position of the hand over time to recognise dynamic gestures. While we use data from the RGB-D sensor for hand detection, the gesture recognition is based on a modified version of the approach presented in (Wobbrock, Wilson, and Li 2007). The system performs reliably well in our target scenarios with an accuracy of up to 90 %.

Deploying CAESAR in Complex Missions

So far, we sketched control modules and the respective underlying methods on the lower level of CAESAR's software architecture and described its human-machine interaction modules. While these modules are inevitable for exhibiting intelligent behaviour of a domestic robot, high-level control entities are necessary for composing the overall behaviour and making the robot act purposefully and goal-directed over longer periods of time without a human operator.

Behavior Engine. There is a large gap between low-level robot control (real-time control and data processing) and high-level task coordination (mission planning, execution monitoring). In our system we bridge this gap using the Lua-based Behavior Engine (BE). It provides skills to the high-level system. Skills are the only means by which the high-level instructs the low-level system and they appear as a basic action to the task planner and encapsulate parameter estimation, communication, concurrency, and failure resilience. They only make local decisions, e.g. whether the object to grasp from the table is reachable, but not global ones like moving to another place to be able to reach it. This is the task of the high-level control. While the skills can become complex, their limitation to local decisions make them easy to handle and re-use as they encode fewer assumptions. Reactive behaviours are modelled as hybrid state machines and are described in detail in (Niemueller, Ferrein, and Lakemeyer 2010). Skills can call other skills hierarchically, allowing to create increasingly complex skills. Skills have also been used to compose simple agent programs, i.e. macro actions that actually do make some global decisions. This has been done for one as a performance comparison for plans from the high-level system. For another, it decreases the number of actions that need to be considered during task planning and therefore reduces the computational burden. These features in particular—hiding low-level details from the high-level system in skills, hierarchical composition, and local failure resilience on the mid-level—make the BE a powerful foundation for our high-level control system.

High-Level Control. For encoding our high-level missions, we developed the robot programming and plan language READYLOG (Ferrein and Lakemeyer 2008). READYLOG is an extension of the well-known action programming language GOLOG (Levesque et al. 1997) which is based on the situation calculus (McCarthy 1963). The situation calculus is a second order logic for reasoning about actions and change. Roughly, the user has to specify a basic action theory (BAT) which describes what properties of the world can be changed by which action, the precondition and effects of these actions and how the world looks like initially. Having a BAT in place, properties of the world can be formally proven. This is in particular useful for planning robot missions. The semantics of READYLOG is defined as situation calculus formulas, as is GOLOG's. Now,

READYLOG integrates features such as dealing with continuous change or probabilistically projecting robot programs into the future among others, which were developed in other GOLOG dialects. In particular, we made use of an extension which integrated decision-theoretic planning. Defining an optimisation theory in addition to the BAT together with non-deterministic choices of actions allows to let the robot plan and evaluate different action alternatives by itself. This way, encoding the behaviour of the robot becomes particularly natural for the programmer. For qualitative fluents, we introduced a semantics based on fuzzy sets, to be able to express and interpret qualitative statements such as "left at the far end of the room". To make READYLOG an effective tool, its system architecture comes with execution monitoring facilities and, for run-time reasons, possibilities to progress the database holding the initial facts. As we only want to overview our system here, we leave aside the details referring to (Ferrein and Lakemeyer 2008; Schiffer, Ferrein, and Lakemeyer 2012b; 2012a).

Lessons Learnt

As the tasks for a domestic robot are manifold, the software architecture used on the robot should allow for integrating third-party software easily. This is especially useful for modules that are out of the scope of one's own research but which are still needed for the whole system to work. Many research software packages in robotics are Open Source. Therefore, it is, in principle, possible to integrate them into one's own robot software. However, there are several issues that make this unnecessarily hard. For one, often such packages cannot be seen as black boxes as they require in-depth knowledge to use them efficiently. Also, integrating new software versions of a third-party module is cumbersome. For another, different levels of abstraction or missing data and information yields incompatibility with the existing infrastructure and makes it hard or even impossible to use foreign components with justifiable effort. Besides that, certain fundamental design decisions may prohibit mixing software from different frameworks. When developing software modules, one should have portability in mind.

After several re-design phases of our own middleware and functional components, we ended up in a hybrid system, where parts of the old software were ported to the new architecture, some routines relied on the old middleware, some on the new one. This made it in particular hard to keep the software system consistent. Of course, the field of software engineering offers proper approaches to circumvent this, but it is a matter of time and man-power to bring these concepts to life. Often, the available resources are restricted and hence quick hacks that were made under pressure stay in the software. This is surely not a new insight. Most of the time goes into software integration. Moreover, it is frustrating enough having to write software that is not in one's research focus, let alone spending much effort in beautifying or just maintaining those sideline components.

As a consequence, we identified a flexible middleware and software framework as a key concern, as it has major influence on system integration. While integration always takes time, a bad framework can drastically increase that very time, or even cause it to fail. *Flexible* means that it provides features that foster re-use of existing components. But it also means, that components developed for a different middleware and framework can be used. It turns out that more often than not, we ended up using more than one such framework. Robot platforms like the Nao or Robotino come with their own framework that must be used to communicate with the underlying hardware. Sometimes software components can be so closely intertwined with a particular middleware that it is harder to decouple function and communication rather than to integrate two different types of middleware. On CAESAR, the high-level system is integrated on top of Fawkes. But underlying, some components and visualisation tools are used from ROS (Quigley et al. 2009), a widely used middleware and framework that gained a lot of attention in the recent years. While we do think that our software provides certain benefits, in particular when it comes to an embodied AI system, we must embrace the rich ecosystem that has evolved around ROS to benefit from it. Hence, close integration of Fawkes and ROS is a major concern.

Integration works both ways. Not only do we want to benefit from the wealth of software that exists. But we also want to share our code and take it to the ROS market place. Therefore, we ported some of our components to ROS, e.g. the Behavior Engine to use it on CMU's HERB (Srinivasa et al. 2012). We gained valuable insights, especially when bringing back changes from the ROS port to our own system. It is now much easier to share code among implementations.

As described earlier, we created our own implementation of a Monte Carlo localization. After years of successful use, we eventually decided to replace it with another one based on the same methods, but implemented—and maintained—by the ROS community. Even though it initially meant integration work and learning about its use, in the long term it promises to save precious resources for primary research.

The overall performance of the system depends not only on the performance of the individual components, but how well they are integrated, if they can efficiently communicate, and if they fit ones overall goals and paradigms, like deliberative high-level control making use of data introspection features—reasons for which we started some of our redesigns. However, working on a new software framework did not come for free and the transition from one framework to another tied resources and did not always go as smooth as planned.

Apart from evaluating and testing the single components of our system individually it is very important to assess the complete system also. All in all, developing a system for a complex target scenario like it is aimed for in RoboCup@Home proves to be a workable approach. Naturalistic settings with tests oriented at real-life situations where laymen operate the robot foster flexible and robust solutions. This is required for the resulting system to be applicable in real-world settings later on.

Discussion

With the experiences made over the years and in light of the lessons learnt that we sketched in this paper we now try to answer the key questions raised at the Symposium.

Q1: *How do we evaluate integrative research?* We think that a competition-based benchmark initiative such as RoboCup@Home is a good way to evaluate integrative research and the resulting complex systems. However, developing every part of such a complex system is cumbersome and a very lengthy process (given only bounded resources). This hinders progress in the overall field. If there would be more exchange and proper interface definitions for exchange on various levels, i.e. ranging from data to sub-components, every group could focus on their particular field of expertise.

Q2: *What are good challenges and benchmark problems?* We believe it is a good idea to target complex, only loosely specified scenarios to force the solutions to be flexible and robust. The final target, i.e. the real world, will very likely not follow any of the simplifying assumptions made at development time. Thus aiming at realistic or at least naturalistic scenarios like in RoboCup@Home for the Domestic Service Robotics domain enforces systems that are more likely to work in real-world scenarios also.

Q3: *What are good platforms and frameworks for this work?* Apart from looking for other systems that follow similar principles as one would have opted for oneself it is not to be underestimated to follow the de facto standard, if only for practical reasons. Sometimes for specific applications, it might still be a good option to use a custom framework, like we opted for a blackboard-driven approach with data introspection for better integration with the high-level system. Working and integrating with a common framework has benefits for both sides of the mutual relationship: module providers get evaluation from the outside (i.e. other groups) and are forced to make their component more robust and flexible, working on a broader ranger of hardware. The component user can save effort in development by using existing (and if mature enough also well-tested) software for problems he/she does not want to care about.

As a conclusion, for the future we plan to increase the compatibility of Fawkes and ROS as the most prominent framework in robotics and the de facto standard. This means integrating ROS components where this is useful, but also to provide means to share our work more easily with the ROS community. Portability among platforms and frameworks should be a key concern for new components. Turnkey packages are an important factor to successfully participate in the domestic service robotics community.

References

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Calmes, L.; Wagner, H.; Schiffer, S.; and Lakemeyer, G. 2007. Combining sound localization and laser-based object recognition. In *Proc. AAAI-SS 2007*, 1–6.

Diankov, R., and Kuffner, J. 2008. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Carnegie Mellon University.

Dylla, F.; Ferrein, A.; Lakemeyer, G.; Murray, J.; Obst, O.; Röfer, T.; Schiffer, S.; Stolzenburg, F.; Visser, U.; and Wagner, T. 2008. Approaching a formal soccer theory from behaviour specifications in robotic soccer. In Dabnicki, P., and Baca, A., eds., *Computer Science and Sports*. WIT Press.

Ferrein, A., and Lakemeyer, G. 2008. Logic-based robot control in highly dynamic domains. *Robotics and Autonomous Systems* 56(11):980–991.

Huang, X.; Alleva, F.; Hon, H.-W.; Hwang, M.-Y.; and Rosenfeld, R. 1993. The SPHINX-II speech recognition system: an overview. *Comput Speech Lang* 7(2):137–148.

Levesque, H.; Reiter, R.; Lésperance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *J Logic Program* 31(1-3):59–83.

Lienhart, R., and Maydt, J. 2002. An extended set of haar-like features for rapid object detection. In *Proc. ICIIP-02*, 900–903. IEEE Press.

McCarthy, J. 1963. Situations, Actions and Causal Laws. Technical report, Stanford University.

Niemueller, T.; Ferrein, A.; Beck, D.; and Lakemeyer, G. 2010. Design Principles of the Component-Based Robot Software Framework Fawkes. In *Proc. SIMPAR-10*, 300–311. Springer.

Niemueller, T.; Ferrein, A.; and Lakemeyer, G. 2010. A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In *RoboCup 2009: Robot Soccer World Cup XIII*, 240–251. Springer.

Quigley, M.; Conley, K.; Gerkey, B. P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Schiffer, S.; Ferrein, A.; and Lakemeyer, G. 2006. Football is coming home. In *Proc. PCAR-06*, 39–50.

Schiffer, S.; Ferrein, A.; and Lakemeyer, G. 2012a. Caesar: An Intelligent Domestic Service Robot. *Journal of Intelligent Service Robotics* 5(4):259–273.

Schiffer, S.; Ferrein, A.; and Lakemeyer, G. 2012b. Reasoning with qualitative positional information for domestic domains in the situation calculus. *Journal of Intelligent & Robotic Systems* 66(1-2):273–300.

Srinivasa, S. S.; Berenson, D.; Cakmak, M.; Collet, A.; Dogar, M. R.; Dragan, A. D.; Knepper, R. A.; Niemueller, T.; Strabala, K.; Vande Weghe, M.; and Ziegler, J. 2012. HERB 2.0: Lessons Learned From Developing a Mobile Manipulator for the Home. *Proceedings of the IEEE* 100(8).

Thrun, S.; Fox, D.; Burgard, W.; and Dellaert, F. 2001. Robust Monte Carlo localization for mobile robots. *Artif. Intell.* 128(1-2):99–141.

Viola, P. A., and Jones, M. J. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proc. CVPR*, 511–518.

Wisspeintner, T.; van der Zant, T.; Iocchi, L.; and Schiffer, S. 2009. RoboCup@Home: Scientific Competition and Benchmarking for Domestic Service Robots. *Interaction Studies* 10(3):392–426.

Wobbrock, J. O.; Wilson, A. D.; and Li, Y. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. UIST-07*, 159–168.