robOCD: Robotic Order Cups Demo – An Interactive Domestic Service Robotics Demo

Stefan Schiffer, Tobias Baumgartner, Daniel Beck, Bahram Maleki-Fard, Tim Niemüller, Christoph Schwering, and Gerhard Lakemeyer

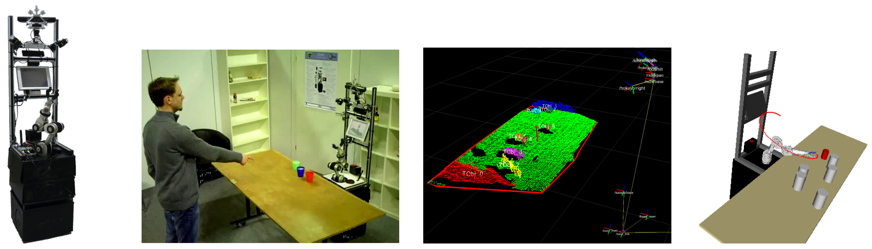
Knowledge-based Systems Group,
RWTH Aachen University, Aachen, Germany
`(schiffer,gerhard)@kbsg.rwth-aachen.de`

Abstract. This paper describes an interactive demonstration by the AllemaniACs' domestic service robot Caesar. In a home-like environment Caesar's task is to help setting the table. Besides basic capabilities of an autonomous mobile robot it uses methods for human-robot interaction and it also has a sophisticated high-level control that allows for decision-theoretic planning. We use this demo to illustrate the interplay of several modules of our robot control software in carrying out complex tasks. The overall system allows to perform robust reliable service robotics in domestic settings like in the RoboCup@Home league. Also, we show how our high-level programming language provides a powerful framework for agent behavior specification that can be beneficially deployed for service robotic applications. The system was showcased repeatedly, most notably at a national RoboCup competition and at an international conference.

1 Introduction

In RoboCup, apart from making agents and robots play soccer, there are also competitions in rescue scenarios as well as in a domestic service robotics setting. While in the soccer leagues quick decision making, cooperation, and team-play are crucial, the challenge in the RoboCup@Home league is to build a robust robotic system that can safely operate in a human environment and that can interact with humans. As the complexity of the tasks to solve in domestic settings rises, so increases the benefit a robot has from using sophisticated means for decision-making and deliberation. The high-level control of Caesar is based on the language Readylog [3], a variant of the logic-based language Golog [4] which combines explicit agent programming as in imperative languages with the possibility to reasons about actions and their effects.

In this paper, we present a demo application of Readylog in a domestic setting to showcase its benefits and its applicability. After we briefly introduce our robot we sketch the domestic service robotics domain. Then we present the robotic order cups demo before we conclude.



(a) Caesar (b) User pointing at a cup (c) Scene perception (d) Manipulation

Fig. 1. Our robot Caesar, the robOCD scenario with a user giving instructions and perception and motion planning in simulation for manipulation in an extended setup.

2 The AllemaniACs Domestic Service Robot Caesar

An increasingly popular application domain for autonomous mobile robots is domestic service robotics (DSR) where robots perform assistive tasks in a home environment. A competition that focuses on these kinds of applications is RoboCup@Home [9]. Apart from the demands that are commonly put on autonomous mobile robots, a service robot in a domestic setting must meet additional requirements with respect to interactivity, robustness, and accessibility. What is more, it can be more helpful in complex assistive tasks if it also features a sophisticated high-level control.

Our robot Caesar, shown in Figure 1(a), is designed and was built to operate in human-populated environments in domestic scenarios. It should be helpful around the house, assisting elderly or disabled people with their daily activities. Caesar meets all the basic requirements put on an autonomous mobile robot, that is, it can navigate in its environment safely and it can localize itself reliably with high accuracy in known environments. Further it is able to detect and recognize people and objects and it can also manipulate objects with its robotic arm. Of particular importance for the demo we discuss in this paper are its robust speech recognition [2] and a component for gesture recognition [8]. Those components are orchestrated in the Fawkes robot framework [5] to form a robust assistive robotic system.

Above all the low-level components mentioned so far and a mid-level Lua-based behavior engine for basic skills of the robot [6], Caesar has a logic-based high-level control that allows for deliberation and flexible decision-making. We use Readylog [3], a dialect of the robot programming and plan language Golog [4]. Golog is based on Reiter's version of the situation calculus [7] which is a sorted second-order language to reason about dynamic systems with actions and situations. Readylog features several extensions to the original Golog language, most notably it allows for decision-theoretic planning in the spirit of DTGolog [1]. On Caesar we use an implementation of a Readylog interpreter in ECL^{i}PS^e-CLP,[1] a Prolog dialect.

[1] Website at `http://www.eclipseclp.org/`

Algorithm 1: Readylog program for the Order Cups Demonstration The terms p_1 to p_4 denote four positions on the table, while I_i and P_i are variables that hold the color of a cup at position i in the initial and the goal situation, respectively. $pos(C)$ returns the position of the cup with color C.

```
proc $main$,
  $get\_Initial\_Order(I_1, I_2, I_3, Init)$;        %% perceive initial order
  $get\_Goal\_Order(P_1, P_2, P_3, Goal)$;          %% inquire about goal order
  $sort\_cups(P_1, P_2, P_3, 4)$;                   %% start planner
endproc
proc $sort\_cups(P_1, P_2, P_3, H)$,
  solve($H, reward\_cup(P_1, P_2, P_3)$,
    while($\neg(p1 = pos(P_1) \wedge p_2 = pos(P_2) \wedge p_3 = pos(P_3))$) do
      pickBest($cup, \{red, green, blue\}$,
        pickBest($to, \{p_1, p_2, p_3, p_4\}, move\_cup(cup, pos(cup), to)$))
    endwhile
  endsolve
endproc
```

3 Robotic Order Cups Demo

We now discuss a special helping task that Caesar is able to perform: the *Robotic Order Cups Demo (robOCD)*.[2] The robot's task in this demo is to help decorating a table. In the scenario there are three differently colored cups (red, green, and blue) on a table. To complete the re-arranging they have to be put in a specific order. A human user is instructing the robot on the desired order by pointing to positions on the table and by simultaneously specifying which cup should be placed at that very position using speech. Figure 1(b) shows a user specifying the desired order of the cups by pointing.

Alg. 1 shows the Readylog procedures used in the demo. The procedure *main* is a sequential program calling sub-procedures for specific tasks. The first step is a call to *get_Initial_Order* to perceive the initial order of the cups on the desk. That is, the robot uses its vision system to detect and recognize three cups, namely one cup for each of the colors red, green, and blue. It stores the initial order of the cups in variables I_i. The call to *get_Goal_Order* then initiates an interactive procedure where the robot asks the user to specify the desired goal positions for the three cups. To do so, the user is requested to point at a position on the table and to say which cup should be placed at that position.

For this to work Caesar's modules for speech and gesture recognition are constantly running. They post the results of their recognition along with a timestamp to a central blackboard. All other modules and the high-level control can access the information there. In the robOCD scenario, the system uses the simultaneous occurrence of position keywords like *there* in the speech recognition

[2] A video of the demonstration is available at `http://goo.gl/7rEEY`.

Algorithm 2: The simplified READYLOG policy for an example run. The initial order was "green, blue, red", the desired order is "red, green, blue".

```
1 $exogf\_Update$, if $\neg done$ then
2    $move\_cup$(blue, $cup\_position$(blue), $p_4$),
3    $exogf\_Update$, if $\neg done$ then
4       $move\_cup$(green, $cup\_position$(green), $p_2$),
5       $exogf\_Update$, if $\neg done$ then
6          $move\_cup$(red, $cup\_position$(red), $p_1$),
7          $exogf\_Update$, if $\neg done$ then
8             $move\_cup$(blue, $cup\_position$(blue), $p_3$)
9 done
```

data and a pointing gesture in the gesture recognition output to determine the desired goal position of a specific cup.

Apart from constructs known from imperative programming languages, e.g., if-then-else, loops, and procedures, READYLOG also offers less common constructs like **solve** and **pickBest**. The former initiates decision-theoretic planning, the latter is the non-deterministic choice of argument. During planning, the logical specification of the dynamics in the world can be used to reason about the state of the world after executing a program. The non-determinism is resolved by opting for those choices that maximize a reward function. For details we refer the interested reader to [3]. Once CAESAR has collected all the necessary information as described above, it determines an execution strategy for the non-deterministic procedure *sort_cups* such that the desired arrangement of the cups is achieved eventually. This is done by means of the decision-theoretic planning just mentioned. In our scenario, the reward function only considers the number of actions needed to reach the desired goal situation. Therefore, CAESAR computes the re-ordering with a minimum number of movements.

The outcome of the decision-theoretic planning is a so-called policy, i.e. a conditional READYLOG program that contains the optimal course of action to achieve the goal. Alg. 2 shows the simplified policy for an example run of robOCD. In the policy, positions p_1–p_3 are the initial positions of the three cups while position p_4 is used as a temporary spot in re-ordering the cups. The method *exogf_Update* is used to update the robot's world model to account for changes which were not due to the robot's actions. The current position of the cup with a specific color is retrieved again every time with the helper function *cup_position*. The initial order of the cups in the example was "green, blue, red". The desired final order is "red, green, blue". The optimal re-ordering, i.e. the re-ordering with the minimum number of actions consists of four *move_cup* actions. To achieve the goal, the robot needs to first move the blue cup to the spare position, then move the green cup to the second spot. After this, the red cup is moved to the first spot and finally the blue cup can be put at the third position. This yields the final order.

The application described above shows the potential benefit of deliberation and that it can be integrated in the behavior specification of a robot very easily.

The high-level control with its decision-theoretic optimization capabilities could be used for path-planning or for more complicated tasks such as planning the course of actions in daily activities of an elderly person. In an extended version of the demonstration the robot drives around looking for different tables with cups on them, picks up cups from those tables and moves them from one table to another. Then, the robot also uses its localization and navigation capabilities to get around and remember table positions. The different versions of the system were successfully showcased repeatedly, most notably at a national RoboCup competition and later at an international conference.

4 Conclusions

In this paper, we presented an interactive demonstration where our domestic service robot Caesar orders cups on a table. The demo integrates methods for robust human-robot interaction like speech and gesture recognition with a logic-based high-level control. The robot can perform a complex task in a domestic setting where it benefits from its deliberation capabilities by using decision-theoretic planning to determine an optimal course of action. Although the demo application of sorting cups seems simplistic at first glance, the system is extendable to more sophisticated tasks with only little effort.

References

1. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: Proc. of the 17th Nat'l Conf. on Artificial Intelligence (AAAI'00). pp. 355–362. Menlo Park, CA (July 2000)
2. Doostdar, M., Schiffer, S., Lakemeyer, G.: Robust speech recognition for service robotics applications. In: Proc. of the Int'l RoboCup Symposium 2008 (RoboCup 2008). LNCS, vol. 5399, pp. 1–12. Springer (2008)
3. Ferrein, A., Lakemeyer, G.: Logic-based robot control in highly dynamic domains. Robotics and Autonomous Systems 56(11), 980–991 (2008)
4. Levesque, H.J., Reiter, R., Lespérance, Y., Lin, F., Scherl, R.B.: Golog: A logic programming language for dynamic domains. J Logic Program 31(1-3), 59–84 (April-June 1997)
5. Niemueller, T., Ferrein, A., Beck, D., Lakemeyer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes. In: Proc. Int'l Conf. on Simulation, Modeling, and Programming for Autonomous Robots. Springer (2010)
6. Niemueller, T., Ferrein, A., Lakemeyer, G.: A Lua-based Behavior Engine for Controlling the Humanoid Robot Nao. In: Proc. Int'l RoboCup Symposium 2009 (RoboCup 2009). LNCS, vol. 5949. Springer (2009)
7. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. The MIT Press (2001)
8. Schiffer, S., Baumgartner, T., Lakemeyer, G.: A modular approach to gesture recognition for interaction with a domestic service robot. In: Proc. Int'l Conf. on Intelligent Robotics and Applications. pp. 348–357. LNCS, Springer (2011)
9. Wisspeintner, T., van der Zant, T., Iocchi, L., Schiffer, S.: Robocup@home: Scientific Competition and Benchmarking for Domestic Service Robots. Interaction Studies. Special Issue on Robots in the Wild 10(3), 392–426 (2009)