Evaluation Metrics for an Experience-based Mobile Artificial Cognitive System*

Liwei Zhang[1], Sebastian Rockel[1], Alessandro Saffiotti[2], Federico Pecora[2], Lothar Hotz[3], Zhenli Lu[4], Denis Klimentjew[1], Jianwei Zhang[1]

Abstract— **In this paper, an experience based mobile artificial cognitive system architecture is briefly described and adopted by a PR2 service robot for the purpose of carrying out tasks within the EU-FP7 funded project RACE. To measure the benefit of learning from experience to improve the robustness of the robot's behavior, an FIM (Fitness to Ideal Model) and a DLen (Description Length) based evaluation approach has been developed. In a restaurant domain where the robot conducts typical tasks of a waiter, a serve coffee to a guest and a clean table task have been implemented to demonstrate the system and the former one is being used in this work to apply the proposed evaluation metrics, validate it on a real robot system and present the evaluation results. They support the co-relation of a reduced fitness when decreasing the description length. This work presents a sound way to evaluate cognitive systems which may take benefit of learning to increase their performance.**

I. INTRODUCTION

The ability to conceptualize stored experiences and to adapt plans and behavior according to experiences is clearly a desirable asset of intelligent robots. It can help robots to expand their knowledge about a complex world, to adapt to changes, and to cope with new situations. Unfortunately, in current robot architectures, experience-based learning has mainly been realized at sub-symbolic levels. RACE (Robustness by Autonomous Competence Enhancement) faces this problem. Nevertheless having an even more advanced system raises the question of how to evaluate this as traditional methods may not be sufficient to do this. There is still a lack of systematic method for performance of using knowledge and experience. In this work, we give an overview of the current state of evaluating complex systems and the evaluation results of the RACE cognitive system.

The main goal of RACE is to develop a framework and methods for learning from experiences in order to facilitate an cognitive intelligent system. A detailed account on what an experience is in computer science as well as how it is represented within RACE is given in section III. To achieve this goal, experiences are recorded as semantic spatio-temporal structures connecting high-level representations, including tasks and behaviors, via their constituents at lower levels down to the sensory and actuator level. In this way, experiences provide a detailed account of how the robot has achieved past goals or how it has failed, and what sensory events have accompanied the activities.

The work in RACE therefore combines research from several communities: (1) an ontology-based multi-level knowledge-representation framework has been devised connecting actuator and sensory experiences with higher-level semantic structures, (2) reasoning facilities for both symbolic and quantitative knowledge are in place to deal with hybrid and diverse knowledge, (3) scene descriptions are enriched by high-level semantic interpretations, (4) learning procedures are being devised exploiting all levels of recorded experiences, and – last, but not least – (5) all this is integrated with a state-of-the-art robot platform (PR2).

The experimental and evaluation domain used in RACE is a restaurant environment where the robot serves guests as a waiter. A typical task may be is to fetch a mug from a counter, bring it to a table and place it in front of the guest on the table. After several experiences, the robot learns how to deal with guests at tables and positions not encountered before. The robot has to deal with different categories of objects known to the robot, i.e. tables, chairs, mugs etc. Furthermore it has to deal with humans which can involve the robot encountering previously unknown objects such as things left on the table like mobile phones, books etc. and the robot has to be prepared with obstacles in his way while driving such as chairs or guests and other waiters passing its way. As such the presented system has to deal with an open-ended world.

Based on the integrated system this work focuses on a new benchmarking and evaluation approach for intelligent robotics which is applied to the described system which integrates the generalization and learning framework developed in the context of the EU project RACE to measure its performance. The presented approach combines a FIM and DLen [1] founded metric applied to multiple runs of scenarios. The results show a strong co-relation between the two. The used definition of FIM is loosely based on [2].

In the next section, we present the architecture of the integrated robot system. The robot memory is a central component for storing multi-level factual knowledge in a coherent way, in particular experiences comprising instructions, plans, activities, perceptions, and failures. The OWL-based ontology structure is designed to connect to a constraint system for temporal, spatial and resource reasoning. The

*This work was supported by the EC Seventh Framework Program theme FP7-ICT-2011-7, grant agreement no. 287752. [1]L. Zhang, S. Rockel, D. Klimentjew, J. Zhang are with the Department of Informatics, University of Hamburg, Germany {lzhang, rockel,klimentj,zhang}@informatik.uni-hamburg.de
[2]A. Saffiotti, F. Pecora are with the AASS Cognitive Robotic Systems Lab, Örebro University, Sweden {asaffio,fpa}@aass.oru.se
[3]L. Hotz is with the HITeC e.V., University of Hamburg, Germany hotz@informatik.uni-hamburg.de
[4]Z. Lu is with the Institute of Electronics and Telematics Engineering of Aveiro, University of Aveiro, Portugal zhenli.lu@ua.pt

representation of concrete scenarios and experience setup are provided in details. Finally, the proposed evaluation approach including a well defined metric is demonstrated on simulation environment and the physical robot platform.

II. RELATED WORK

Artificial Cognitive Systems (ACS), as compared in [3], enabled autonomous robots not only to realize fast and robust motion control in the dynamic environment, examples are [4]–[7], but also to learn from experience and to improve its performance [8]. In order to evaluate the performance of behavior spaces of the autonomous robot, Mali and Mukerjee [9] firstly defined a behavior metric. A detailed theoretical analysis together with the "WashDish" task carried out by a robot simulator was discussed to illustrate the significance of the proposed metrics.

The International Electrotechnical Commission (IEC) published a performance evaluation method for applications in a household-like environment [10]. In this performance evaluation method, the authors aim at formulating general standard terms for robot capabilities and how to measure them. The methods are bound to a service robots in a household environment. Many performance terms like pose and carrying capability are defined. But there is no evaluation metrics for knowledge and experience based systems since these robot platforms for household usually have no learning facility.

RoboCup@Home [11] is the largest international annual competition for autonomous domestic service robots. It designs several benchmark tests where the robots' abilities and performance are tested in a realistic non-standardized home environment setting. A statistical analysis of the benchmarks from recent years was also given. However, most of these tests focus on physical capabilities, the evaluation of high-level learning and abstract capabilities is missing.

The Performance Metrics for Intelligent Systems (PerMIS) aims towards defining measures and methodologies for evaluating the performance of intelligent systems. It focuses on robotic applications concerning practical problems with respect to performance measures. The book [12] consists of results collected from the past workshops. It includes a collection of related work rather than defining a common benchmark.

In the scientific field of mobile manipulation, the "Mobile Manipulation Challenge" aims to provide a snapshot of the state of the art autonomous mobile manipulation applications. In 2010, the challenge focus on a constrained pick-and-place task, e.g., object retrieval, loading a dishwasher. The ICRA 2012 Mobile Manipulation Challenge is similar: clearing a table, setting a table, and serving from a rotating table in a "sushi boat" restaurant environment. The ICRA 2013 Mobile Manipulation Challenge held in a kitchen environment, where the robots can operate the objects in the kitchen. Two tasks are assigned and performance is evaluated by scoring. This challenge focuses on manipulating objects in the kitchen, like cutlery and bowels. It is unconcerned with knowledge and experience based systems.

```
!Fluent
Class_Instance: [MoveArmToSide, moveArmToSide1]
StartTime: [61.142, 61.142]
FinishTime: [66.306, 66.306]
Properties:
- [hasArm, RobotArm, leftArm]
- [hasResult, ActivityResult, succeeded]
```

Fig. 1. An example Fluent.

In this emerging and challenging field of performance evaluation, an important development has been made. However, until now, little research has been done about evaluating the experience-based artificial cognitive system with the service behaviors of a real mobile robot. There is currently no accepted standard for quantitatively measuring the performance of using a knowledge base and experiences in an ACS. In this paper, we propose an evaluation approach which bases on FIM and DLen metrics. The former method is derived from the work of [13].

III. RACE ARCHITECTURE

In this section, we present the main components of the modular RACE architecture and describe important features of the knowledge representation and reasoning (KR&R) framework. The components have been implemented as ROS packages and integrate to a running system on the physical and simulated PR2 robot platform.

The basic data type is a `Fluent`. `Fluents` are exchanged through ROS messages with the blackboard. All `Fluents` are instances of concepts from the ontology. An example Fluent is depicted in Fig. 1. Its semantics is a robot activity or an event and can also represent an experience. A `Fluent` consists of a concept from the ontology, two time intervals for start and end and can have certain properties. This work references the term `Experience` frequently and defines it as follows [14]:

Definition An experience in this work is defined as an execution trace that integrates subsymbolic and symbolic representations and provides a detailed account of success or failure.

The central component of the RACE system is the `Blackboard` as depicted in the architecture (Fig. 2). Its contents can be seen analogous to what can be found in an ABox in Description Logics. Its main goal is, like in traditional blackboard architectures, to keep track of updates of other system components. Basically, the robot's internal and external sensed state is reflected here for further processing. This is done by other modules for perception, reasoning, planning and execution in order to eventually write their results back into the `Blackboard`. It will be exploited in the RACE system to maintain a consistent representation of `Fluents` (executed actions, world state propositions, etc.), that include a begin and end timestamp each and to store a complete experience record (e.g., execution of a coffee serving task) to be conceptualized later. From the `Fluents`, the current state and the past state information can be derived.

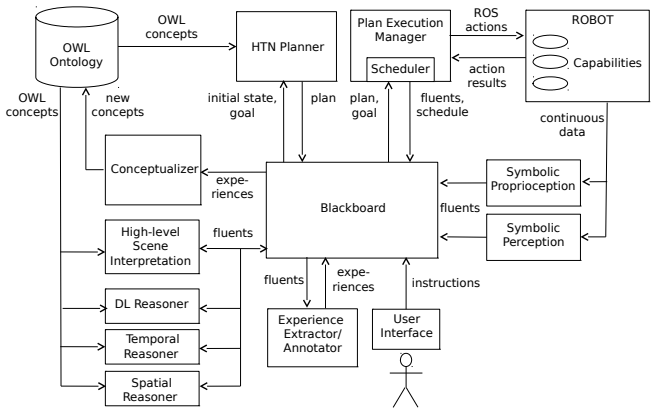

Fig. 2. The RACE Architecture contains a Blackboard as the robot's central memory. Other components mainly communicate with it in order to exchange and process Fluents.

Details of the experience representation format can be found in [14].

Through the user interface a planning goal is entered into the blackboard which triggers the `HTN` (Hierarchical Task Network) Planner [15]. The Planner creates then its initial planning state and writes a generated plan back to the Blackboard. Pre- and post-conditions to operators within the plan and a hierarchy of expanded HTN methods are included. The plan triggers furthermore the `Plan Execution Manager` which has the task to dispatch the planned actions and execute the appropriate robot actions. Before any robot capabilities, e.g. grasping, are triggered the `Scheduler` will paralyze robot actions where possible according to a resource based approach. Robot actions at this level are executed at a certain level of reactivity when necessary. While executing, the `Plan Execution Manager` will log the success and failure information at planning operator level to the `Blackboard`.

Within this project a PR2 robot, running ROS (Robot Operating System) [16] is being used. Basic capabilities, e.g. for manipulation or navigation, are already part of ROS and can be accessed in the form of ROS actions. Others needed within the RACE system have been added. Robot capabilities are tightly coupled to planning operators in the planning domain specification. Thus, an operator can be executed directly. Typically, capabilities run in a tight perception-action loop and some failure cases can be handled locally. Nevertheless, if an action permanently fails this is escalated to the `Plan Execution Manager`, which will in turn trigger re-planning.

Not all plan failures can be detected locally by the capabilities itself. For instance, if a bottle is picked up, but slips from the gripper on the way to another table, the pick-up action has already terminated successfully. However, the plan execution manager can infer that the bottle has slipped by using proprioception (since the gripper fully closes). By checking the current world state against the operator preconditions, this kind of plan failure can be detected. More generally, the detection of failures requires inference by which the robot's own expectations are checked for consistency with respect to the observed state of the world. In RACE, this is achieved by invoking ontological and constraint-based reasoning services (described in [14]).

The robot provides continuous data about its own status (such as joint angles) as well as data from its various sensors. Then, the `symbolic perception/proprioception` modules discretize this information into symbolic timestamped `Fluents`; example outputs of these modules indicate whether the robot's gripper is open, closed, or moving, or the qualitative spatial location of a piece of cutlery observed by the robot's RGB-D camera. While the outputs refer to discrete symbolic entities (such as "servingArea1"), these entities may have quantitative properties (such as the spatial polygon that "servingArea1" represents).

The robot's conceptual knowledge is stored in the `OWL Ontology` [14]. The ontology provides a common representation format from which the domain knowledge of all other reasoners is generated. For instance, the HTN planning domain is extracted from the conceptual knowledge about robot activities. Similarly, the `OWL Ontology` feeds the spatial, temporal and ontological reasoners as well as the high-level scene interpretation. These reasoners enrich the basic experiences in the `Blackboard` with higher-level semantic information and can also be queried directly by other modules. For instance, if the robot's task is to serve coffees, temporal inference may tell it that the act of delivering coffees must be scheduled before the coffees become cold. Similarly, metric spatial reasoning can endow the robot with the ability to discern exactly where to place a dish when serving (between the fork and the knife of a well-set table).

Background processes, responsible for experience extraction and conceptualization support a long-term learning loop. The history of `Fluents` in the Blackboard is continuously observed by the `Experience Extractor` to detect and extract potential relevant experiences, based on plan structure, user feedback, pauses, similarity with stored experiences, and conceptualizations and novelty.

Experiences are then used to improve future performance by employing several learning methods. Taking the experiences from the `Blackboard` as input, the `Conceptualizer` [14] modifies the ontology, resulting in more robust and flexible future plans.

IV. EVALUATION APPROACH

To measure success for a given task in a given scenario, we use an approach inspired by model-based validation techniques [13]; namely, we measure the compliance of the actual robot's behavior to the intended ideal behavior for that task in that scenario. Fig. 3 graphically illustrates this principle: the trace of a given execution of the RACE system is compared against a specification of what the ideal behavior should be, resulting in a "Fitness to Ideal Model" (FIM) measure. These specifications will be formulated in a way

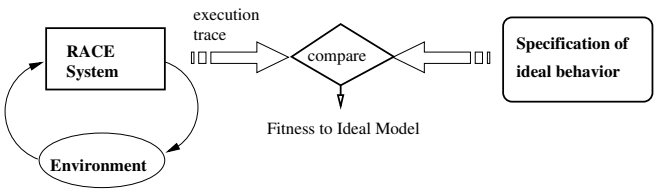

Fig. 3. Principle of evaluation in RACE: the system's behavior is compared to a model of the ideal behavior for the specific scenario.

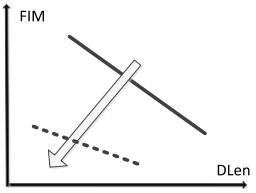

Fig. 4. Overall view of the RACE aim: to develop tools for autonomously learning knowledge that allows to specify the robot's task by as few as possible instructions (low DLen) to achieve correct behavior (low FIM).

that facilitates the task of automatically computing the FIM measure, as discussed in [17] and presented below.

Discrepancies between the observed behavior and the ideal behavior can originate from errors of four different types:

- **Conceptual errors** — e.g., the robot places a mug outside of the guest's reach because it does not know that all objects should be served within the guest's placing area.
- **Perceptual errors** — e.g., the robot fails in perceiving a mug.
- **Navigation** and/or **localization errors** — e.g., the robot places a mug on the wrong table because it is wrongly localized.
- **Manipulation errors** — e.g., the robot fails to pick up a mug from the table because it slips from the gripper.

The latter three types of errors – perceptual, navigation and manipulation errors – are platform specific. They do not indicate problems with the intended behavior of the robot, but with its physical execution. As the RACE project addresses the learning and use of knowledge for increasing the robot performance, our metrics mainly focus on quantifying conceptual errors.

Conceptual errors arise from discrepancies between the knowledge used by the robot and the one encoded in the specification of the ideal behavior. We call these discrepancy *inconsistencies*. Specifically, inconsistencies can be of four types:

- **Temporal inconsistencies**, that is, inconsistencies that are due to not adhering to a temporal constraint — e.g., the robot fails to serve coffee within a given deadline.
- **Spatial inconsistencies**, that is, inconsistencies caused by not adhering to a spatial constraint — e.g., the robot places a mug on the wrong side of the table.
- **Taxonomical inconsistencies**, that is, inconsistencies that derive from a wrong conceptual taxonomy — e.g., the robot serves wine in a coffee mug rather than a wine glass.
- **Compositional inconsistencies**, that is, inconsistencies deriving from the lack of causal support and/or wrong hierarchical decomposition — e.g., the robot does not clear all mugs from a table.

Accordingly, we will adopt the following four metrics to quantify performance of the robot in relevant tasks:

$$p_t = \tau_t \cdot \#\text{temporal\_inconsistencies}$$
$$p_s = \tau_s \cdot \#\text{spatial\_inconsistencies}$$
$$p_x = \tau_x \cdot \#\text{taxonomical\_inconsistencies}$$
$$p_c = \tau_c \cdot \#\text{compositional\_inconsistencies}$$

where $\tau_{(\cdot)} \in [0, 1]$ are weights which determine the importance of the four types of inconsistency. Together, the four above define the FIM metric:

$$\text{FIM} = \sum_{i \in \{t,s,x,c\}} p_i \tag{1}$$

If the ideal robot behavior is specified using a formal model that allows to "count" inconsistencies, then this FIM metric is operational. If that formal model even allows to detecting and counting inconsistencies automatically, then the FIM metric can be computed automatically. In the case of temporal inconsistencies, for instance, we will employ consistency checking procedures in temporal constraint networks representing the temporal aspect of ideal robot behavior — e.g., a Fluent [18] representing the task of serving coffee and a temporal constraint representing the deadline for serving the coffee can be checked against the execution trace of the robot through simple temporal constraint propagation procedures. Clearly, other formalisms could be used to specify the ideal behavior [19]. In this project, we will preferably use the formalisms which are used to represent `Fluents` inside the RACE system, since we expect that having the same formalism inside the system and on the external reference model will be convenient when evaluating the effect of learning.

In addition to estimating the effectiveness of learned knowledge by counting the number of inconsistencies, we are also interested in measuring the `Description Length` (DLen) of the instructions that should be given to the robot to achieve a goal. It is important to include this dimension into the evaluation, as flawless behavior can always be achieved by over-specifying the task, e.g. tele-operating the robot. We conjecture that the FIM measure for a given task in a given scenario will proportionally decrease with the description length of the instructions increasing, as shown by the solid line in Fig. 4. Successful behavior following shorter

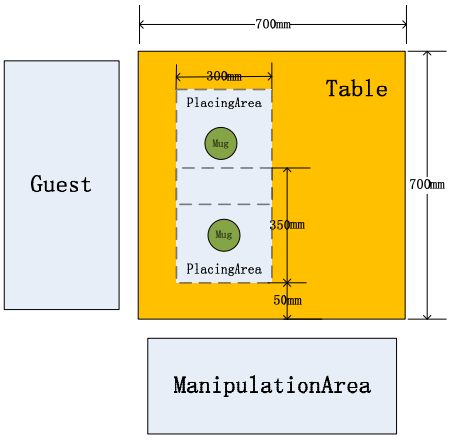

Fig. 5. PlacingArea and ManipulationArea of the table1

instruction descriptions is indicative of the effectiveness of the learned knowledge. Also, this may indirectly provide a measure of how general the knowledge is if applied to a wide range of scenarios and initial conditions.

Overall, our aim in RACE is to develop a system of learning and reasoning tools that will allow the robot to autonomously and effectively increase its competence. This overall aim is related to the FIM and DLen metrics as indicated in Fig. 4, which summarizes the final objective of RACE: to make long specifications unnecessary for the achievement of highly fitting behavior (i.e. behavior which generates few temporal, spatial, taxonomical and compositional inconsistencies). Graphically, this increase in competence is indicated by the transition from the solid line to the dash line.

V. SCENARIO SET-UP AND EXPERIMENTS

We will demonstrate robot behavior in an experimental restaurant domain. To collect experiences, the robot will carry out tasks of a waiter, bringing food and beverages to any of several tables, placing dishes properly in front of guests, clearing tables, etc. In this work, two demonstrations named "ServeACoffee" and "ClearTable" have been defined and performed on the physical PR2 platform in a restaurant environment. The results are presented and evaluated with respect to the metrics defined in Section IV and described in [17], [20].

A. Scenario Set-up

In the restaurant environment, the robot must transport food and drinks to a specified area. Hence, we predefine the PlacingArea. The PlacingArea is the part of the table where mugs and dishes should be placed. It is a rectangle area with length of 350 mm and width of 300 mm. The distance from the edge of the table to the PlacingArea is 50 mm. The mugs, dishes and spoons can be placed anywhere in the PlacingArea. The PlacingArea of the counter is slightly different. The PlacingArea definition of table is shown in Fig. 5.

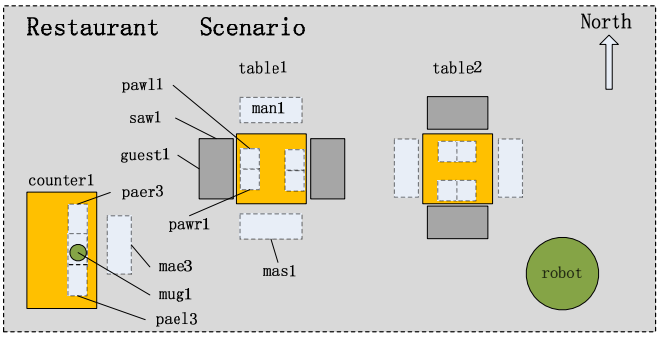

Fig. 6. Initial floor plan of the restaurant environment

We first present three scenarios in the restaurant domain which will be used for the "ServeACoffee". The idea is to let the robot discover a generalization of the "ServeGuest" experiences in Scenarios A and B which will subsume the task in Scenario C. Then two scenarios are set up for the demonstration "ClearTable". In the first scenario, the robot is taught to clear the table by several instructions. After learning the experience of "ClearTable", the robot will execute the "ClearTable" activity by one instruction.

B. ServeACoffee Demonstration

Scenario A: The restaurant floor plan is as shown in Fig. 6, where the robot knows the position (pae) of mug1 on counter1, the position of table1, the position of guest1 west of table1, and the regions for manipulation, sitting and placing. The user successively instructs the robot to move to counter1, grasp mug1, move to the manipulation region (mas1) south of table1, and place mug1 at the placement region (pawr1) west of table1. The robot is told that this is a "ServeGuest" activity.

Scenario B: The same as Scenario A, except a new guest (guest2) is sitting east of table1 and the robot is instructed to move to the north of table1 and place mug1 at the east of table1. Again, the robot is told that this is a "ServeGuest" activity.

Scenario C: Guest3 is sitting south of table2 and the robot is simply instructed: Do a "ServeGuest" to guest3.

Schedule for "ServeACoffee" scenario A is list as follows. There are 13 steps to accomplish the task. The 4 concrete instructions are included in the schedule.

1) `robot` gets the environment model with world coordinates, including coordinates of corresponding areas (like instances of `PlacingAreaSouth`) and initializes its own position (`position1`). The guest sits at west of `table1`.
2) `user1` instructs `robot` to move to `mae3`, the eastern manipulation area of `counter1`.
3) `robot` plans a sequence of actions with an HTN planner, based on a pre-existing model, to reach `mae3`. This involves actions `tuck_arms`, `move_torso` and `move_base`.
4) `robot` moves to `mae3` as planned.

5) user1 instructs robot to grasp mug1 from pae3, the eastern placing area of the counter1.
6) robot plans a sequence of actions with an HTN planner, based on a pre-existing model, to grasp mug1 from pae3. This involves actions move_torso, tuck_arms, move_arm_to_side and pickup_object.
7) robot grasps mug1 from pae3 as planned.
8) user1 instructs robot to move to mas1, the southern manipulation area of table1.
9) robot plans a sequence of actions with an HTN planner, based on a pre-existing model, to reach mas1. This involves actions move_base_blind, move_torso and move_base.
10) robot moves to mas1 as planned, reaching the destination still holding mug1.
11) user1 instructs robot to place mug1 in pawr1, the right western placing area of table1, in front of guest1.
12) robot plans a sequence of actions with an HTN planner, based on a pre-existing model, to place mug1 in pawr1. This involves actions move_torso, move_base_blind and place_object.
13) robot (successfully) places mug1 in pawr1, as planned.

C. ClearTable Demonstration

Scenario A: The robot knows the position of mug1 and mug2 on the table1 (east of the table1), the position of counter1. The robot is instructed to move to table1, grasp mug1, move to the counter, and place mug1 on the counter1 and then move mug2 from table1 to counter1. The robot is told that this is a "ClearTable" activity.

Scenario B: The same as Scenario A, except mug1 and mug2 are placed on the opposite sides of the table and the robot is instructed to clear the table1 and to place mug1 on the counter. Again, the robot is told that this is a "ClearTable" activity.

D. Experimental Results

In this section, the evaluation metrics described in section IV will be applied to the two demonstrations. To save space, only "ServeACoffee" results are shown. The results of "ClearTable" are similar.

Let $V0$ be the nominal (ideal) condition of the demonstrator (as described in the scenario A schedule). In $V1$ (as described in scenarios B and C), the guest sits on the opposite side of the table (or leaves the sitting area), other than specified in the planning domain. Robot still brings the mug in the same place as before (which is now in front of an empty seat). Then the compositional inconsistency (the replanning is needed but not included in the demonstrator) and the perception error (the guest perception is needed but not included) occur. That means:

$$\begin{aligned} \text{\#spatial\_inconsistencies} &= 1 \\ \text{\#compositional\_inconsistencies} &= 1 \end{aligned}$$

According to the evaluation metrics defined in the last section, $\tau_{(\cdot)}$ is the weight which determines the importance of the this type of inconsistency. Here we set weight $\tau_{(\cdot)} = 1$. The initial value of the four types of inconsistency is assigned to be 0. Then we have

$$\begin{aligned} FIM(V0) &= 0 \\ FIM(V1) &= \text{\#spatial\_inconsistencies} \\ &\quad + \text{\#compositional\_inconsistencies} = 2 \end{aligned} \tag{2}$$

The evaluation has been tested on the three scenarios of "ServeACoffee", respectively. In scenario A, 50 experiments have been executed to obtain the experimental results. Some indicators like move to mae3 have been checked from all the experiments. In Tab. 1, the plan steps are checked with respect to Scenario A. All the results are judged by human. There are 43 times executed successfully, which are shown in the first column of Tab. 1 (only the first time experiment result is list). The second column shows the result of the 8th experiment, where the robot failed to grasp the mug1 because the grasping solution is not feasible. The third column shows the result of the 13th experiment, where the robot fail to detect the mug1 on the counter. The fourth column shows the result of the 27th experiment, where the robot fail to place the mug1 in the specified area. The fifth column shows the result of the 42nd experiment, where the robot fail to move to the mae3 because of the navigation system error. The latter 4 columns show the errors caused by other types of errors such as **Perceptual errors** and **Navigation errors**, which will not be counted in the **Conceptual errors**.

In Scenarios B and C, the corresponding item of place mug1 in pawr1 is Failure. The reason of Failure is that the guest moves to a position at the table other than the one specified at the start. But the robot would bring the mug to the old position.

Fig. 7 shows the statistical results of all kinds errors occurring in three scenarios. In scenario A, there are 43 times successful execution and 7 errors (2 Perceptual errors, 2 Navigation errors and 3 Manipulation errors). In scenario B, there are 40 times successful execution and 10 errors (3 Perceptual errors, 5 Navigation errors and 2 Manipulation errors). In scenario C, there are 37 times successful execution and 13 errors (6 Perceptual errors, 4 Navigation errors and 3 Manipulation errors). All the errors are judged by human during the process of the experiments.

To measure the description length of the instructions given to the robot, step by step instructions are provided to the robot. In scenarios A and B, a set of instructions were provided. In the following instructions list of Scenario A, each achieve command specifies a sub-task to be carried by the robot and represents an instruction. The last teach command is the instruction to teach a new task. It is a

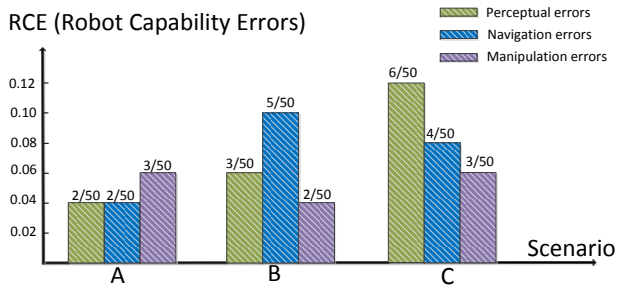

Fig. 7. RCE (Robot Capability Errors) in the ServeACoffee scenario

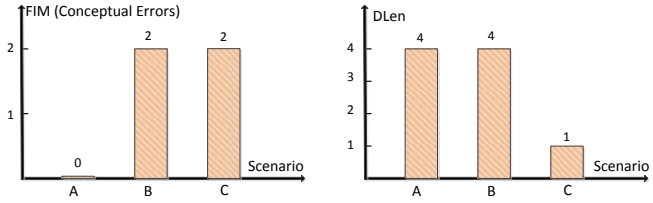

Fig. 8. Conceptual errors (left) and the Description Length of the instructions in ServeACoffee (right)

composition of the given set of sub-tasks, as shown in the following:

1) achieve drive_robot_Task preManipulationAreaEastCounter1
2) achieve grasp_object_w_arm_Task mug1 rightArm1
3) achieve drive_robot_Task preManipulationAreaSouthTable1
4) achieve put_object_Task mug1 placingAreaWestRightTable1
5) teach_task ServeACoffee guest1

In Scenario B, similar instructions were provided by the user. In Scenario C, only a single `achieve` instruction is provided as follows. Now the robot can execute the "ServeACoffee" task with a shorter instruction set:

1) achieve serve_coffee_to_guest_Task guest3

Fig. 8 shows there are 4, 4 and 1 instructions in three scenarios of "ServeACoffee", respectively. Fig. 9 shows the relationship between the FIM and DLen. The yellow circle dot (4,0) and (1,2) means that there are 4 instructions and 0 FIM errors, 1 instruction and 2 FIM errors in the scenario A and B of "ServeACoffee", respectively; while the blue square (3,0) and (1,3) means that there are 3 instructions and 0 FIM errors, 1 instruction and 3 FIM errors in the scenario A and B

Scenario A	Instr.	A: ex 1	A: ex 8	A: ex 13	A: ex 27	A: ex 42
move to mae3	1	Success	Success	Success	Success	Failure
detect mug1 on pae3	2	Success	Success	Failure	Success	/
grasp mug1 from pae3	2	Success	Failure	/	Success	/
move to mas1	3	Success	/	/	Success	/
place mug1 in pawr1	4	Success	/	/	Failure	/

TABLE 1
EXPERIMENTAL RESULTS OF SCENARIO A

of "ClearTable", respectively. We find that the execution will close to the ideal situation when the instructions increase. Theoretically, the best situation will be the point (1,0), which means there is no inconsistency when the robot triggered by one instruction.

The restaurant environment is shown in Fig. 10. In Fig. 11, the PR2 robot tries to grasp mug1 from counter1 during Scenario A. In Fig. 12, mug1 has been placed in pawr1. The video attached to this paper shows the execution process of the Scenario A. The scenarios might be executed in the physical or the simulated environment, as indicated by the figures.

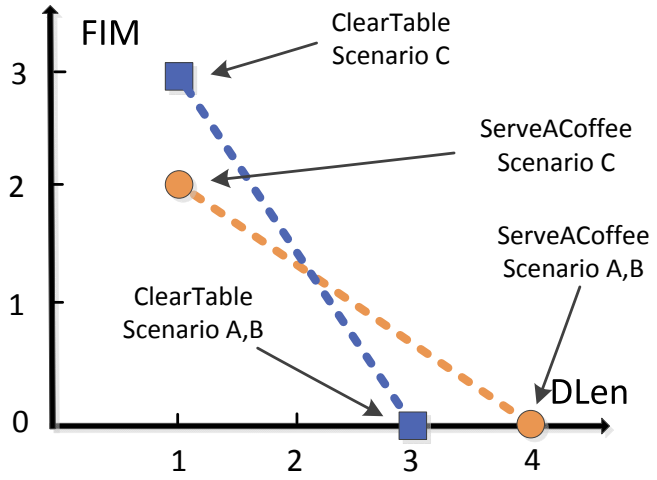

Fig. 9. The relationship between FIM and DLen

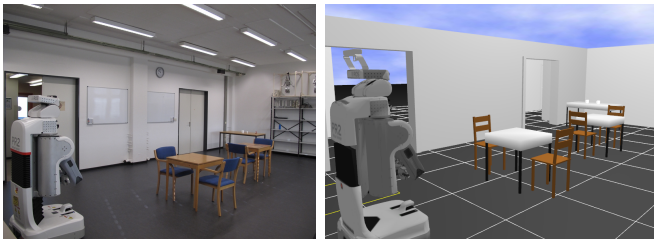

Fig. 10. The restaurant environment in a typical start condition: the robot waits for a guest and to be instructed

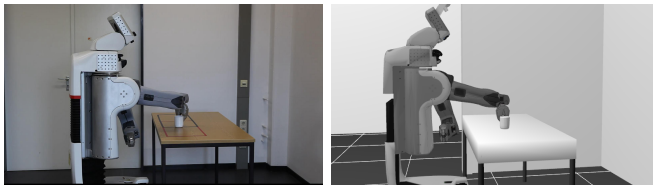

Fig. 11. In order to serve a guest the robot grasps a mug from the counter in Scenario A

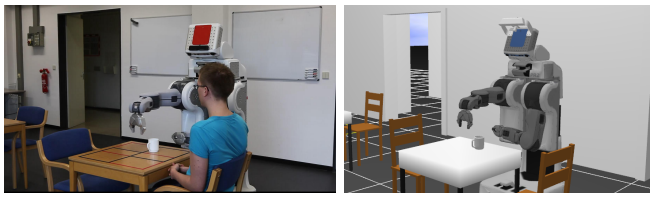

Fig. 12. The mug is placed in front of a guest. This involves (learned) concepts of serving the guest from the right side and doing it on a convenient place for the guest in (Scenario A)

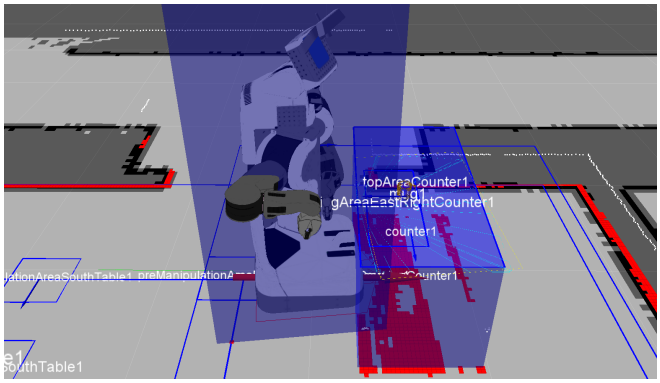

Fig. 13. Visualization of the robot's knowledge (blackboard contents) about its environment. Physical entities have a 3D bounding box and and non-physical, e.g. a placing area on a table, a 2D bounding box.

VI. CONCLUSIONS AND FUTURE WORK

In this work, an experience-based cognitive intelligent system has been introduced to execute scenarios in an open-ended world. In order to measure its performance an evaluation approach was proposed. The FIM and DLen model were adopted to measure the compliance of the actual robot's behavior to the intended ideal behavior for a given task and scenario, i.e. the demonstrations of "ServeACoffee" and "ClearTable". The proposed artificial cognitive system has been evaluated with the defined metrics and the results presented. The data obtained indicates an improvement in the robot's knowledge and behavior thus the metric is appropriate to evaluate such a system. The initial assumption in an FIM and DLen co-relation is supported by the evaluation results.

Future work on new scenarios will show how the benchmark copes with different tasks, new objects etc. and will evaluate the benchmark itself. Work has to be done on judging errors in an automated manner. This will contribute to gain more test results by running many more scenarios and will lead to an even more significant conclusions. Another aspect worth investigating is the amount of Fluents processed or created within a specific task which can be collected automatically and added to the result data. As the importance of the learning will increase in the future another measure on how the degree of learning improves the performance has to be investigated.

REFERENCES

[1] Peter Grünwald. A tutorial introduction to the minimum description length principle. *CoRR*, math.ST/0406077, 2004.

[2] H. W. Marsh, K. T. Hau, and Z. Wen. In search of golden rules: Comment on hypothesis-testing approaches to setting cutoff values for fit indexes and dangers in overgeneralizing Hu and Bentler's (1999) findings. *Structural Equation Modeling*, 11(3):320–341, 2004.

[3] David Vernon, Giorgio Metta, and Giulio Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation, special issue on Autonomous Mental Development*, 11:151–180, April 2007.

[4] Yushan Chen, Jana Tumova, and Calin Beltav. Ltl robot motion control based on automata learning of environmental dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5177–5182, 2012.

[5] Muhammad Attamimi, Keisuke Ito, Tomoaki Nakamura, and Takayuki Nagai. A planning method for efficient mobile manipulation considering ambiguity. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 965–972, 2012.

[6] KangGeon Kim, Dongkyu Choi, Ji-Yong Lee, Jung-Min Park, and Bum-Jae You. Controlling a humanoid robot in home environment with a cognitive architecture. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1754–1759, 2011.

[7] Sebastian Rockel, Denis Klimentjew, and Jianwei Zhang. A multi-robot platform for mobile robots - a novel evaluation and development approach with multi-agent technology. In *IEEE Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 470–477, 2012.

[8] Liz Murphy, Steven Martin, and Peter Corke. Creating and using probabilistic costmaps from vehicle experience. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4689–4694, 2012.

[9] Amol Dattatraya Mali and Amitabha Mukerjee. Metrics for evaluation of behavior-based robotic systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1122–1127, 1998.

[10] International Electrotechnical Commission (IEC). Performance evaluation method of intelligent mobile robot platform for household and similar applications. Technical report, International Electrotechnical Commission (IEC).

[11] *RoboCup@Home Rules & Regulations*. 2012.

[12] Raj Madhavan, Edward Tunstel, and Elena Messina, editors. *Performance Evaluation and Benchmarking of Intelligent Systems*. Springer, 2009.

[13] P. S. Kaliappan. Model-based verification techniques: State of the art. Technical report, Brandenburg University of Technology, 2008.

[14] S. Rockel, B. Neumann, J. Zhang, K. S. R. Dubba, A. G. Cohn, Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, S. Stock, J. Hertzberg, A. M. Tomé, A. J. Pinho, L. S. Lopes, S. von Riegen, and L. Hotz. An ontology-based multi-level robot architecture for learning from experiences. In *Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium*, Stanford (USA), March 2013.

[15] Dana Nau, Héctor Mu noz Avila, Yue Cao, Amnon Lotem, and Steven Mitchell. Total-order planning with partially ordered subtasks. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, volume 17, pages 425–430, Seattle, 2001.

[16] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[17] Liwei Zhang, Sebastian Rockel, Federico Pecora, Luís Seabra Lopes, Alessandro Saffiotti, and Bernd Neumann. Deliverable d5.1 - evaluation infrastructure. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, November 2012.

[18] Jos Lehmann, Bernd Neumann, Stephanie von Riegen, Lothar Hotz, Masoumeh Mansouri, Federico Pecora, and Sebastian Stock. Deliverable d1.3 - hand-coded non-hybrid knowledge contents. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, November 2012.

[19] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[20] Liwei Zhang and Sebastian Rockel. Deliverable d5.2 - year-1 demonstrator. Technical report, European Commission - Information and Communication Technologies - Seventh Framework Programme, January 2013.