A Cognitive Architecture for Automatic Gardening

Alejandro Agostini[1,*], Guillem Alenyà[2], Andreas Fischbach[3], Hanno Scharr[3], Florentin Wörgötter[1], and Carme Torras[2]

[1]*Bernstein Center for Computational Neuroscience, Göttingen, Germany.*

[2]*Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Barcelona, Spain.*

[3]*Institute of Bio- and Geosciences: Plant Sciences (IBG-2), Forschungszentrum Jülich GmbH, Jülich, Germany*

Keywords: cognitive architecture, optimized plant treatments, automatic gardening, human-robot interaction, learning planning operators.

Abstract

In large industrial greenhouses, plants are usually treated following well established protocols for watering, nutrients, and shading/light. While this is practical for the automation of the process, it does not tap the full potential for optimal plant treatment. To more efficiently grow plants, specific treatments according to the plant individual needs should be applied. Experienced human gardeners are very good at treating plants individually. Unfortunately, hiring a crew of gardeners to carry out this task in large greenhouses is not cost effective. In this work we present a cognitive system that integrates artificial intelligence (AI) techniques for decision-making with robotics techniques for sensing and acting to autonomously treat plants using a real-robot platform. Artificial intelligence techniques are used to decide the amount of water and nutrients each plant needs according to the history of the plant. Robotic techniques for sensing measure plant attributes (e.g. leaves) from visual information using 3D model representations. These attributes are used by the AI system to make decisions about the treatment to apply. Acting techniques execute robot movements to supply the plants with the specified amount of water and nutrients.

1 Introduction

While agriculture tasks entailing repetitive action on the basis of immediate sensing have been successfully automated [1], gardening considering longer time frames has remained elusive. Taking individualized care of plants is difficult because although plants react to changes of light, water and/or nutrients, these reactions come with long, variable delays and are also quite variable in strength. Moreover, individual reactions

*Corresponding author. *E-mail address*: aagosti@gwdg.de (A. Agostini).

are history dependent, because an initially healthy plant will react differently to a treatment than a stressed or damaged one.

Human gardeners take actions relying on their *expertise* and their knowledge of the *history of events* observed in a given plantation. However, to attend to the needs of a plantation in large industrial greenhouses, having a team of gardeners might not be cost effective. The alternative is to use a robotic system capable of taking actions according to the specific needs of every plant. This is the main focus of this paper, where a robotic cognitive architecture for automatic gardening is presented. The architecture integrates artificial intelligence techniques for decision-making with robotics techniques for sensing and acting. Low-level robotic approaches comprises short-term interaction of the robot with plants entailing 3D model acquisition of deformable objects (leaves) [2] and robot-arm manipulation approaches for active vision and individual treatment application [3].

For making decisions we implemented the decision-making framework presented in [4], specifically designed to generate action plans in tasks involving long delays between actions and effects. This framework focuses on the cognitive abilities needed, namely decision-making based on logic-based planning and learning weakly correlated cause-effects along sequences of events. These mechanisms compile expertise so as to deduce, from past sequences of events, the long-term treatments producing the best desired results in each particular situation of a plant. Planning and learning are interleaved in a way that permits uninterrupted operation by requesting a human gardener to instruct the treatments when the knowledge acquired so far is insufficient to make a decision. In this case, the gardener simply specifies the immediate treatment to apply to the plant while the learning approach autonomously finds the relevant events that compactly describe the plant evolution under that treatment. For instance, the gardener may instruct the robot to provide small doses of nutrient in combination with doses of water three times a day, if he detects a low growth rate of the plant due to lack of nutrients, or simply small doses of water twice a day, if he knows that large doses of nutrients and water has been administered right before. With these instructions, the system progressively improves the decision-making performance by coding plant evolutions into planning operators until the human intervention is not longer required.

Many industrial and service tasks require these cognitive abilities, and we have worked on the gardening setting because of our involvement in the European project GARNICS [5]. The goal of GARNICS was to automatically monitor large botanic experiments in a greenhouse to determine the best treatments (watering, nutrients, light) to optimize predefined aspects (growth, plant status) and to eventually guide robots to obtain the required measures from leaves and apply the treatments prescribed by the system.

The cognitive architecture has been tested in an industrially-relevant plant phenotyping application [6, 7], with very encouraging results as presented in Section 3. In the final experiment within the GARNICS project, images of Tobacco plants (Nicotiana tobacum cv. Samsun) were acquired every hour for up to 30 days, from which the required appearance features were extracted. Moreover, for each individual plant, light intensity was measured, and water and nutrients were dispensed using an automated flexible-tube pump. The image dataset has been made publicly available [8, 9] and further details on how these experimental data were collected can be found therein (cmp.

also Section 3, esp. Figure 6).

1.1 Related Works

Robotics applications for the execution of human-like tasks have been tackled using techniques of human-robot interaction, task planning, and symbol grounding [10]. Artificial intelligence planning techniques play a fundamental role in such applications since they use a symbolic, logic-based notation compatible with human language. This allows for a natural human-robot interaction, letting a lay person easily provide instructions to the robot [11]. For a successful integration of these techniques in a robotic platform, it is mandatory to ground the symbolic descriptions of the AI planning methods to let the robot interact with the real world in order to execute a task [12]. This is normally done by integrating methods of different levels of abstractions [10, 13, 14, 15, 16], ranging from purely symbolic methods [17], to sensor information processing [18] and acting methods [19]. Several approaches integrate planning and acting for the robotic execution of human like tasks. In [15], a new architecture that integrates task and motion planning (TMP) is proposed. This architecture uses an incremental constraint-based task planning to dynamically incorporate motion feasibility at the task level during task execution, facilitating the symbol grounding problem. The architecture presented in [20], on the other hand, incorporates human assistance to demonstrate relevant actions for the task. The main novelty of their approach is a method for learning from demonstration that permits reusing knowledge of previously learned tasks to accelerate the learning of new ones. This is done by exploiting semantic similarities between tasks parameters. Learning from demonstration has also been used in the architecture presented in [16]. In this case, a human provides example executions of symbolic actions that are used to update low-level probabilistic models parametrized for each specific robotic platform, which permits evaluating the feasibility of grounding symbolic actions.

Robotics applications has also been implemented for the task of automatic gardening [21, 1, 22]. In [21] a set of mobile robots are equipped with eye-in-hand cameras and mobile arms to water and harvest tomato plants according to their individual needs. To recognize position and maturity level of tomatoes (green or red), the system selects a robot that moves to the plant to take images from six (fixed) different perspectives. Tomatoes are recognized using a feature-based method that detects circles and smooth areas using a convolution approach [23]. A humidity sensor is placed in the soil to evaluate watering requirements. Task planning is carried out to allocate plants to robots. However, they do not apply task planning or any other AI approach to decide the treatment for each plant according to its needs. Water is supplied to the plant in a fixed amount and in a reactive manner when the value of the humidity sensor drops below a predefined threshold. Another example of a robotic system applied to automatic gardening is presented in [22]. In this case, the system is split in two parts: one part attached to the greenhouse for the control of the environmental light, temperature, and humidity, and the other one fixed on a robot to control plant watering and seeding. Watering and seeding take place at predefined positions in the soil and are triggered by manually activated buttons. As in [21], humidity sensors are placed in the soil to reactively water plants using a thresholded approach. In their architecture, none of the automatic

processes uses artificial intelligence techniques for planning or learning.

In the following sections we present our cognitive architecture for automatic gardening. Section 2 introduces the general architecture of the system. In this section the robotic mechanisms for sensing and acting as well as the artificial intelligence techniques for planning and learning are described. The performance of the cognitive architecture in a real scenario is assessed in Section 3 and deeply discussed in Section 4. The paper ends with some conclusions (Section 5).

2 General Architecture of the System

Fig. 1 presents a general scheme of the cognitive architecture. The architecture is composed of a decision-making framework (DMF), in charge of deciding the action (treatment) to be applied to the plant, the modules for perception and action, which ground the symbolic states and actions, respectively, and a KUKA arm robotic platform equipped with a set of cameras and a probe to supply water and nutrients.

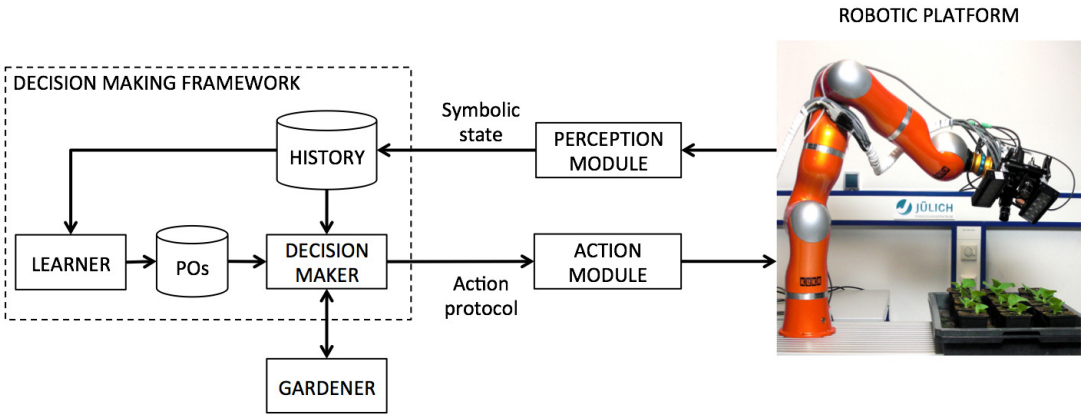

Figure 1: General scheme of the system.

The DMF receives a description of the plant from the *perception module*, which transforms the images captured by the cameras into a symbolic description of the plant state (e.g. number of leaves, size of the plant, etc.). The state description is stored in the *history* database that is used by the *decision maker* to define which treatment to apply depending on the previous history of the plant and the goal specification. The treatment is then sent to the *action module* for its actual execution. The history database is also used to learn which of the past experienced events are actually relevant to predict the evolution of the plant under a treatment. This is the task of the *learner*, which generates and refines *planning operators* (POs) containing these relevant events. The system is supported by a human *gardener* that specifies the treatment to apply when the POs learned so far are not enough to make a decision in the current situation. After the treatment execution, the process starts over again by generating a new plan from the reached plant state. This is a strategy known as replanning, which enables the robot to immediately make use of the new knowledge acquired after the learning step. It also

allows for task continuation from an unexpected effect, that would spoil the previous plan, requiring a new plan to be found. The following sections provide more details about the mechanisms described above.

2.1 Basic Notation

In this section we introduce the domain definition for task planning in automatic gardening. As in most planning approaches the domain definition is handcrafted using the knowledge of a human expert [24, 17]. In our case, we used the expertise of a gardener to define the relevant state and action variables for the task of controlling the growth of plants.

The set of attributes used to describe a plant *state* at a specific time t, s^t, are presented in Table 1. Those attributes referencing size, e.g. mean size of leaves, are described using the standard notation: XS (extra small), S (small), M (medium), L (large), and XL (extra large). All the other attributes are specified using integer values.

Number of leaves (n leaves)
Number of green leaves (n green leaves)
Number of yellow leaves (n yellow leaves)
Ratio green/total leaves (green/total leaves)
Mean size of leaves (mean sz)
Size of smallest leaf (small sz)
Size of largest leaf (large sz)
Plant height (height)
Percentage of mean growth rate of leaves (mean gr)
Percentage of largest leaf growth rate (large gr)

Table 1: Plant attributes used in the state description.

In a similar way, we define an *action* (or *treatment*), a^t, applied to the plant at a specific time t, as a sequence of doses of water and nutrient supplied to the plant in intervals of 1 hour during 24 hours. A treatment may also involve variations in the light intensity along this period. Tables 2 and 3 present two examples of treatments of *strong nutrient* and *mild watering*, respectively.

Using the specified notation, it is straightforward to represent the control behaviours as cause-effect couples (CECs) in a way easily interpretable by humans. This simplifies the task of the gardener in supervising the plant evolution and specifying which treatment to be applied when needed. Table 4 shows an example of the changes in the number of leaves under the application of two treatment doses of 'Strong Nutrient',

Atomic action	$t+0$	$t+4$	$t+8$	$t+12$	$t+16$	$t+20$	$t+23$
Water (ml)	1	0	1	0	1	0	1
Nutrient (ml)	2	1	2	1	2	1	2
Light variation (%)	0	0	0	0	0	0	0

Table 2: Treatment of strong nutrient.

where relatively large doses of nutrient are supplied to the plant. Each column corresponds to the attributes observed and the treatment applied at each time interval of 24 hours.

A complete description of the cause-effects taking place in the plant should not only consider the changes taking place with an action (Table 4) but also the past events that allow these changes to occur. As pointed out in the Introduction (Section 1), the currently observed evolution of a plant depends not only on its current state and the recently applied treatments, but also on the past history of the plant. An example of this can be observed in Table 5, where a complete description of the causes that would allow the changes coded in Table 4 to actually occur is presented.

The CEC represented in Table 5 indicates that, in order to obtain the coded changes with the execution of two treatments of 'Strong Nutrients', the value 'small sz=XS' should have been observed 24 hours in advance, 'n green leaves=0' and 'height=XS' 48 hours before the treatments execution, and all the variables at time 0 hours should be observed just before the treatments are applied. In addition, the CEC also indicates that two treatments of 'Mild Watering' (Table 3) should have been applied 24 and 48 hours in the past, respectively.

Note that the cause-effect couple in Table 5 provides a compact representation of the evolution of the plant, in the same way a planning operator provides a compact representation of state transitions. In other words, a CEC does not show the complete sequence of state transitions that would be observed under the executed treatments, which would involve all the variables of the state space, but a compact representation of them, only considering the attributes that change with the treatments and those that are relevant to accurately predict these changes.

2.2 Perception Module

The symbolic state representation is generated by the perception module from the images captured by the cameras. Both color and near-infrared (NIR) images are used to measure plant structure. NIR images provide better gray level contrast and the procedure to locate leaf boundaries becomes easier (Fig. 2).

Segmentation and counting of leaves in images is a hard problem and still an ongoing research topic [25]. Tobacco images acquired in our scenario as well as *Arabidopsis* images, both of them accompanied by ground truth segmentations and annotations, are available to the public as a benchmark dataset [9]. While for *Arabidopsis* results steadily improve [26, 27, 28], tobacco leaf segmentation and counting remains challenging [29]. Here, we use a previously developed multi-level procedure. First, a background removal process is applied to remove the pot, the soil, and the rest of the background. Second,

Atomic action	$t+0$	$t+5$	$t+10$	$t+15$	$t+20$
Water (ml)	1	1	1	1	1
Nutrient (ml)	0	0	0	0	0
Light (%)	0	0	0	0	0

Table 3: Treatment of mild watering.

Time	0 h	24 h	48 h
n leaves	2	2	4
n green leaves	2	2	4
mean sz	S	M	M
small sz	S	M	XS
large sz	S	M	L
height	XS	S	S
mean gr	10%	10%	20%
Treatment	Strong Nutrient	Strong Nutrient	

Table 4: Example of Plant Changes.

Time	-48 h	-24 h	0 h	24 h	48 h
n leaves			2	2	4
n green leaves	0		2	2	4
mean sz			S	M	M
small sz		XS	S	M	XS
large sz			S	M	L
height	XS		XS	S	S
mean gr			10%	10%	20%
Treatment	Mild Watering	Mild Watering	Strong Nutrient	Strong Nutrient	

Table 5: Example of Complete Cause-Effect Description.

the leaves are segmented using a superparamagnetic clustering method [30, 31]. Unfortunately, this method produces oversegmentation and leaves can be represented by more than one segment. Third, a correction is performed to merge segments corresponding to the same leaf. This correction is based on the convex hull approximation using depth information, which is derived from the disparity map obtained by stereo matching [2]. The accuracy of the perception system depends on the inclination of the leaf and is measured as the validity (correlation between the measured contour and a leaf model) of the extracted segments. In the fronto-parallel position, the measured validity is about 0.9 and decreases linearly up to 0.5 when the leaf is rotated to 40 °.

Figure 2: NIR image of a single plant.

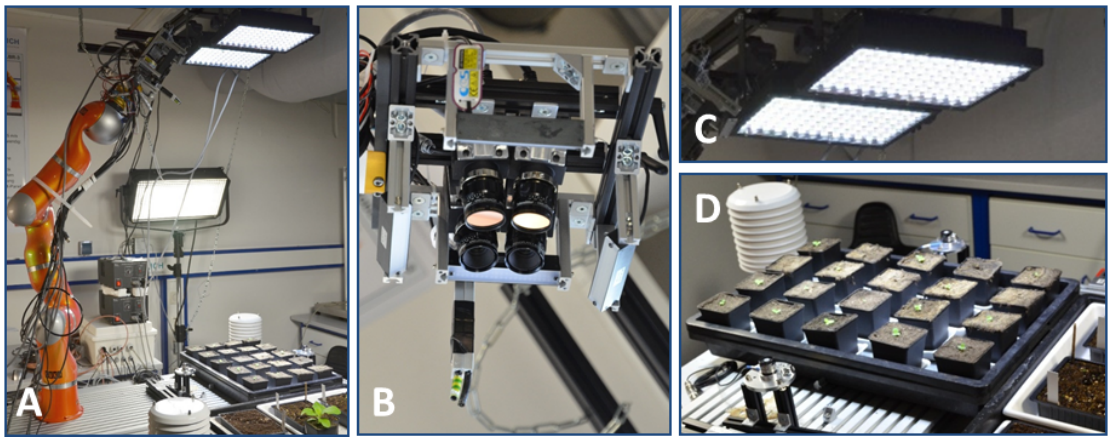

Figure 3: Hardware setup of the robot gardener as used in the experiment: A: the robot arm, in the lab environment. B: Camera head with illumination for imaging and watering system. C: High power LED illumination. D: Workspace with light-, temperature-, and humidity-sensors.

2.3 Action Module

The treatments are applied to individual plants through the action module. The system is designed to treat multiple plants. In current phenotyping facilities [32, 33, 34] plants are transported with conveyor belts or robotic systems to the measurement cabin. Plants may be in individual pots or, as in our case, in a tray with other plants. In our setup, the measurement cabin is substituted by a robot that can actively change the position of the sensors, and also can apply individual treatments [3]. The robot is equipped with two tanks, one with plain water and the other with nutrients dissolved in water. A probe is mounted on the robot hand (Fig, 3B) to administer the treatments by moving the probe to each one of the plants. The treatment includes also the amount of light, that is jointly controlled for the whole set of plants, but individually measured in a calibration step considering illumination inhomogeneities.

2.4 Selecting the Treatment to Apply

For the selection of the treatments to be executed we integrated in the robotic platform the AI decision-making framework (DMF) for executing tasks involving weakly correlated cause-effects presented in [4]. In this section we present a summary of such a framework. For a more thorough description please refer to [4].

The DMF comprises a logic-based planning method, which generates a plan (sequence of treatments) that would bring a plant from the current state to the desired goal state, and a learning approach, interleaved with planning, that automatically generates planning operators coding weakly correlated cause-effects.

Planning

To generate plans, the planning approach uses a set of planning operators that compactly code the changes obtained with the execution of actions. A planning operator is

composed by the precondition, action, and effect parts. The precondition part enumerates the conditions that should be observed to produce the coded changes. The effect part, in turn, describes these changes, observed after the execution of the actions coded in the action part.

In our approach, the compact representation of changes is provided by the cause-effect rules, which are directly transformed into POs using a precondition-action-effect notation compatible with a logic-based planner. To provide an example of how a CEC is coded as a PO, Table 6 shows the PO representation of the CEC described in Table 5, where two treatment doses of 'Strong Nutrient' are released in sequence. All the preconditions that should be observed in the history of the plant to obtain the coded changes, are all the events coded in the CEC taking place in the past up to $t = 0$. [1]

To find a plan from every possible situation, the planner should be provided with a complete set of planning operators that would permit obtaining any desired goal from those situations. Since planning operators are learned from observed executions, the planner may face a situation in which no plan can be generated. In this case, exploratory actions (treatments) should be executed to generate new planning operators to fill that gap of knowledge. To speed up learning, we let a human gardener guide this exploration by instructing the immediate treatments to be applied. This permits quickly generating operators coding relevant cause-effects for the task at hand. Note that the exploration of actions could also be performed automatically (e.g. by executing actions randomly) but this would require a much larger amount of time to generate task-relevant cause-effect rules. Figure 4 presents a snapshot of the graphical user interface (GUI) used by the gardener to instruct treatments.

Learning Weakly Correlated Cause-effects

The evolution of the plant observed after every treatment execution is used by the learning method to find relations between past events and observed changes. If the treatment was instructed by the gardener, the system generates a new CEC from the observed changes that would fill the gap of knowledge that originated the request. If the treatment was provided by the planner but its execution does not produce the expected effects, the system refines the executed CEC with those past events that also need to be considered to accurately predict the expected changes.

The learning approach focuses on two aspects. In the first place, the accuracy of a CEC is evaluated using a probabilistic estimate, called the *density*-estimate [35], that takes into account the lack of experience in the estimation to avoid large biases when the estimation is based on few examples. The formula of the *density*-estimate is

$$\mathcal{P} = \frac{n_+ + \hat{n}_\emptyset \, c}{n_+ + n_- + \hat{n}_\emptyset}, \tag{1}$$

[1] Note that logic-based planning approaches assume a Markovian decision process model of the environment [24], i.e., that the effects depend only on the state observed when the action is executed. This is not the case for the task of controlling plant growth, where the effect of applying a treatment depends on the history of the plant. Therefore, to make our representation compatible with logic-based planning, we used the well-known fact that a non-Markovian decision process can be transformed into a Markovian one by redefining the state representation so as to include history. For more information about this transformation please refer to [4].

Precondition	Action	Effect
n leaves = 2 at 0 h	Strong nutrient at 0 h	n leaves = 2 at 24 h
n green leaves= 2 at 0 h		n green leaves = 2 at 24 h
Mean Sz = S at 0 h		Mean Sz = M at 24 h
Small Sz = S at 0 h		Small Sz = M at 24 h
Large Sz = S at 0 h		Large Sz = M at 24 h
Plant height = XS at 0 h		Plant height = S at 24 h
Mean Gr = 10% at 0 h		Mean Gr = 10% at 24 h
Small Sz = XS at -24 h		
n green leaves = 0 at -48 h		
Plant height = XS at -48 h		
Action mild watering at -24 h		
Action mild watering at -48 h		
	Strong nutrient at 24 h	n leaves = 4 at 48 h
		n green leaves = 4 at 48 h
		Mean Sz = M at 48 h
		Small Sz = XS at 48 h
		Large Sz = L at 48 h
		Plant height = S at 48 h
		Mean Gr = 20% at 48 h

Table 6: Example of the precondition, action, and effect parts of a planning operator (PO) coding the cause-effect rule shown in Table 5. All the events coded in the PO (either attributes or treatments) have associated their time of occurrence. The action part consists of two treatments of 'Strong Nutrient'. The precondition part contains the relevant events that need to be observed in order to accurately predict the evolution of the plant with the execution of the two treatments. The effect part codes the expected evolution of the plant after each treatment execution.

where $\mathcal{P}$ is the CEC probability of obtaining the coded changes with the applied treatments. The variable n_+ is the number of *positive* experienced sequences. These sequences are those that match the past events and treatments considered in the CEC, as well as the coded changes. In turn, n_- is the number of *negative* experienced sequences, which also match the past events and coded treatments but, in this case, do not match the changes coded in the CEC. The variable $\hat{n}_\emptyset$ is the estimated number of sequences that are still pending to be experienced so as to consider the estimation as confident. $\hat{n}_\emptyset$ balances the influence of the experienced instances, mainly when few experiences are gathered, and should progressively decrease to 0 as more experiences are collected. Finally, the parameter c is the *a-priori* probability that the not experienced sequences $\hat{n}_\emptyset$ will be positive once experienced.

The second aspect is how to efficiently find relevant past events, i.e. those events that would increase the probability (1) to obtain the coded changes. While learning proceeds, potentially relevant combinations of events are kept in memory and progressively refined with those past events that produce an increment in the probability. The refinement process of a CEC, say CEC_i, starts by retrieving from the database those

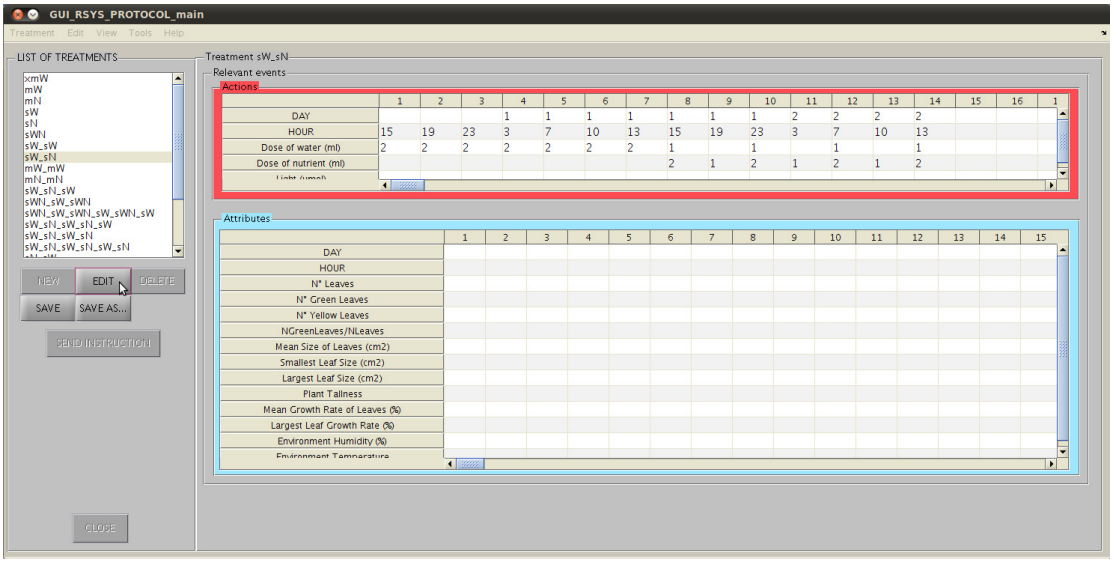

Figure 4: Snapshot of the GUI to instruct treatments. The gardener is able to select, edit, or generate the treatments to apply using the panel at the left of the GUI. The actions involved in the treatments, e.g. doses of water and nutrients, together with the corresponding time-steps are deployed in the editable table Actions at the top-right. The GUI also permits specifying the attributes that would be involved in the cause-effect rule associated to the instructed treatments (Attributes table) but this option was not used during instruction since the system is able to automatically find them.

CECs coding the same changes as CEC_i, and gathering them in the set $\mathcal{R}_{change}$. Then, new combinations of past events are generated in two different ways. On the one hand, for each CEC $\in \mathcal{R}_{change}$, many alternative refinements in one past event, of the ones already considered in the CEC, are performed using events extracted from positive sequences corresponding to the CEC. On the other hand, new combinations are generated from all the possible combinations of the past events of any two CECs $\in \mathcal{R}_{change}$. All the generated CECs, as well as the previously existing ones (i.e. those in $\mathcal{R}_{change}$), are merged together in the set of *candidate* CECs, $\mathcal{R}_{cand}$. Finally, the candidate CEC with highest probability (1) is used to refine CEC_i.

To shed some light on the performance of the mechanisms for learning weakly correlated cause-effects, we show in Fig. 5 the results obtained in several experiments carried out in a simulated plant scenario [4]. The simulated scenario permits obtaining enough data to exhaustively assess the performance of the system in a reasonable amount of time, which otherwise would have taken several months to collect if real plants were used.

The experiments were carried out presenting simulated plants to the system in a sequential way, each of them having a different initial situation, which was generated randomly. The goal for all the plants was to bring them to a large size and with a minimum number of 10 leaves, where at least 80% of them should be green. In the experiments, the logic-based planner PKS was used [36]. The reference for the performance is provided by the OBSERVER method [37], a well-known learning strategy that inter-

leaves the learning of planning operators with logic-based planning. The OBSERVER approach learns planning operators using an adaptation of the Version Space method [38] by keeping and refining the most general and the most specific representation of the precondition part of each operator.

Fig. 5A shows the accumulated number of gardener instructions during learning. At the beginning, when no CECs were generated, the gardener needs to intensively instruct the actions (treatments) to apply. Due to the learning process, the gardener's intervention decreases progressively until the gardener's help is no longer required. The accumulated number of unexpected effects also converges (Fig. 5B), showing how the initially incomplete CECs are successfully refined to include the required past events. Note that the accumulated number of unexpected effects for the OBSERVER does not converge due to its inability to find all the past relevant events that should be considered to successfully obtain the changes obtained from the instructed treatments.

Figures 5C and 5D also demonstrate how the learning progressively finds all the relevant past events for each of the CECs generated. Figure 5D shows the ratio of plans completed without any gardener intervention or unexpected effect. Note that, in the long run, all the plans were completed successfully using our approach. Finally, Figure 5C presents the accumulated number of cause-effect rules generated during learning, showing that the rule generation also ceases in the long run.

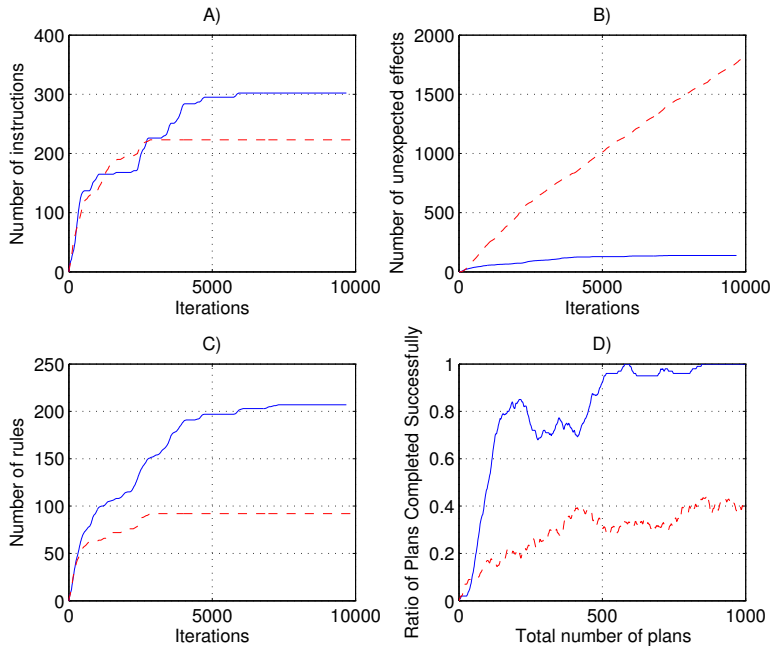

Figure 5: Results obtained from the simulated plant scenario. The continuous blue lines correspond to our cognitive architecture while the dashed red lines correspond to the reference method OBSERVER (source [4]).

Next section presents the performance evaluation of the robotic cognitive architecture for the task of controlling the growth of real plants.

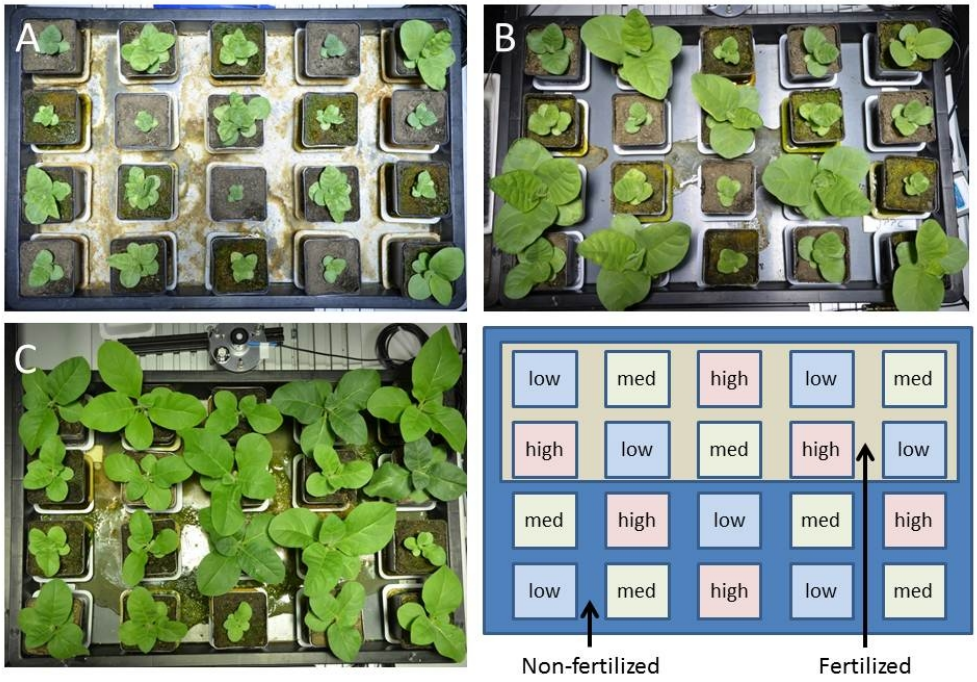

Figure 6: Training experiments. Different treatments are applied to generate new CECs, and consequently some plants are big and some small. Light intensities applied were (A) 100% (i.e. too much light), (B) 50%, and (C) 25% (relatively low light). Treatment plan (D) describes low (0.6ml), medium (1.8ml), or high (3.0ml) amounts of water applied every 2 hours, respectively, and no or 12ml Hakaphos green 0.3% two times per week [8]. The experiments ended after 30 days or when it is not possible to segment contiguous plants anymore.

3 Performance Evaluation

To demonstrate the validity of the cognitive architecture in real scenarios, we present experiments carried out with real plants using the robotic platform of the GARNICS project (see Figure 3). A real experiment consists in treating some plants, possibly in different initial conditions, and grow them up to a desired state. In the real scenario, cause-effect couples (CECs) are learned either offline from training data acquired in previous runs or online in the current experiment. Offline learning is carried out to speed up the initial learning process by generating CECs from sequences collected in advance.

Offline CEC acquisition

The system was trained through 3 experiments of 18, 20, and 30 days of duration (depending on the growth rate due to the treatment) including 20 plants each. Figure 6 shows the 3 plant trays after the experiments, together with the watering and nutrition layout. The treatments for each plant were generated with as much diversity of cause-effects as possible. Observe that some plants grew while others remained very small. Treatments were manually defined by the gardener.

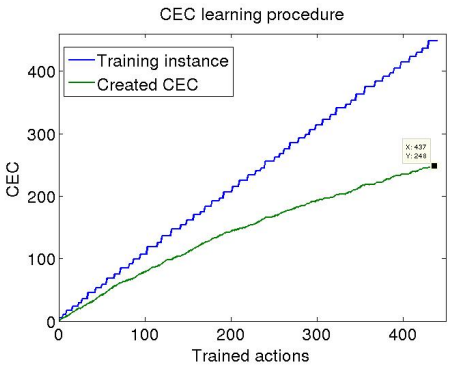

Figure 7: Training experiments: Number of training instances and learned CECs.

Figure 7 shows the evolution of the number of CECs generated during the learning process. A training instance consists of a sequence of states and treatments until a change in the state is observed, which may happen after a single day or after several days of applying treatments. Each new experiment produces several training instances, and consequently their number grows linearly. However, some of these training instances may have produced changes already experienced before, and do not produce a new CEC but are used to refine existing ones. After this process, 248 new CECs were created out of 448 training instances.

Plant caring experiment

The goal of this experiment is to use the real robot to take care of a set of 20 plants, starting from different initial conditions, and to grow them automatically using the previously acquired CEC while requesting teacher assistance when required. The experiment was 12 days long. Fig. 8 shows the input data used to make decisions each day for 4 representative plants, where only the number of leaves and the growth rate were considered for the state description. The state of the plant was updated every hour, and decisions about the treatment once per day. Decision points are marked with a circle. Observe the high variability of the data along the same day due to noise in the perception process.

An identification number was assigned to each plant. Observe in Fig. 8 that all plants, except plant A880643, start the experiment with 0 observable leaves. Table 7 shows the CECs selected for the 4 plants along 5 days. Observe that CEC 1 is selected for all the plants except for plant A880643. CEC 1 involves three treatments executed along 3 days: {Day 1: treatment 1, Day 2: treatment 1, Day 3: treatment 2}. Treatment 1 consisted in pouring 2ml nutrients twice a day, and treatment 2 consisted in no action. The expected change at the end of these treatments is the emergence of one leaf, which should happen on the third day.

During the 3 days, plants A880593 and A880603 behaved as expected, and no change in their state was observed, hence no replanning was needed. Regarding plant A880573, observe in Fig. 8a that the state change predicted by the CEC occurred on the second day, not the third one as expected. An unexpected effect signal was again triggered. These unexpected effects were used to learn and refine the CEC as explained

day	PlantID	CEC	Treatment
1	880573	1	1
	880593	1	1
	880603	1	1
	880643	**11**	**10**
2	880573	1	1
	880593	1	1
	880603	1	1
	880643	11	13
3	**880573**	**27**	**39**
	880593	1	2
	880603	1	2
	880643	20	30
4	880573	13	10
	880593	82	97
	880603	82	97
	880643	20	26
5	880573	60	77
	880593	**56**	**44**
	880603	82	97
	880643	56	44

Table 7: CECs and corresponding treatments selected for 4 plants in the experiment during 5 days (cmp. also Figure 8). Observe that at day 3 plant 880573 has an unexpected behaviour which triggers replanning, while plants 880593 and 880603 finalize the 3-day treatments. For the same plants, the same CEC is selected for day 4, but plant 880593 changes its state unexpectedly, and thus a new CEC is selected. For plant 880643, as its initial state is different (see Fig 8d), the first selected CEC is different.

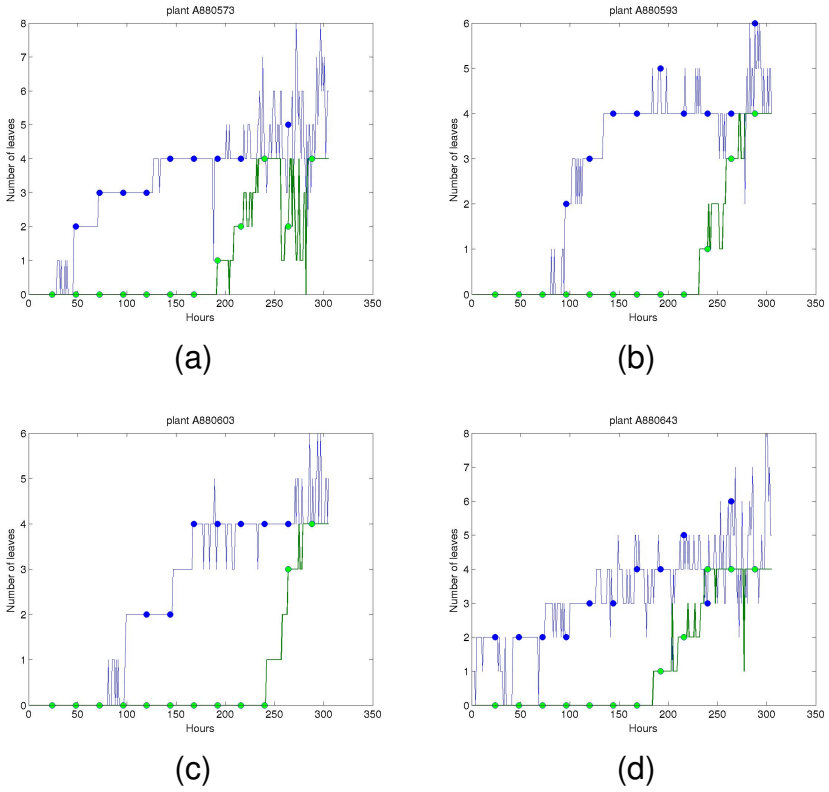

Figure 8: States for 4 different plants: number of leaves in blue, growth rate in green, planning points indicated with marks. Even if treated with the same treatments (first 72 hours), observe the different behavior of the plants.

in Sec. 2.4.

Additionally, observe that on the 4th day the state of plants A880593 and A880603 was the same, and consequently the same CEC was selected (Figs. 8b and 8c). However, on day 5 the CEC execution for plant A880593 produced an unexpected effect, and the planner selected a different CEC for it. Regarding plant A880643, it behaved as expected and 2 different 2-day treatments were applied with no unexpected effect, that is, at the end of each day the state of the plant corresponded to the predicted one (Fig. 8d).

The overall performance in terms of growth rates is shown in Figure 9. We observe, that the optimized treatments applied by our system lead to a significant increase of the mean growth rate in the test experiment, compared to the best-performing training plants, i.e. the top performing plants in training experiment 3. The average growth rate of the 17 healthy test plants is 0.18 (i.e. 51% area increase per day), whereas the best-performing training plants show a growth rate of 0.16 to 0.17 (i.e. 45% to 48% area increase per day).

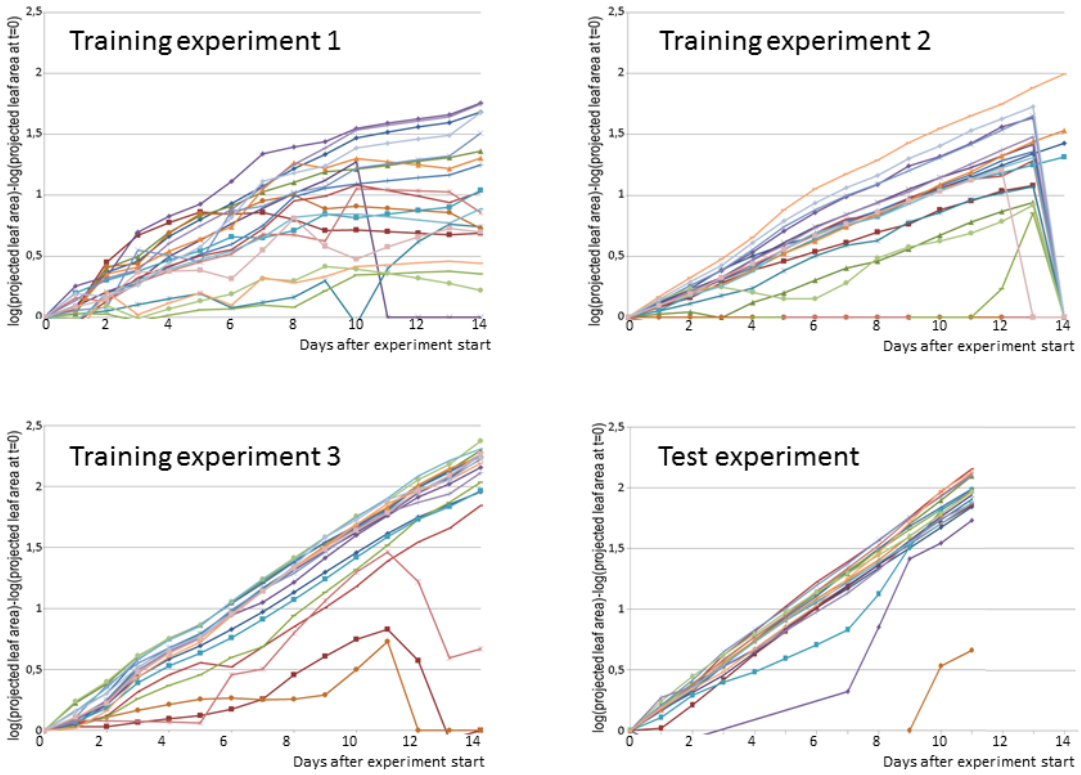

Figure 9: Logarithmic sizes wrt. inital sizes for the training experiments and the test experiment. In this representation growth rates are the inclinations (derivatives) of the graphs. In each plot each color stands for a different individual plant.

4 Discussion

The integration of AI method with robotic mechanisms into a real robot cognitive architecture entails several challenges. One of the biggest challenges is the symbol grounding problem, i.e. filling the gap between artificial intelligence techniques and robotic approaches for sensing and acting. To address the symbol grounding problem we defined a symbolic notation easily transformable into the representation used by the sensing and acting mechanisms. The symbolic variables are obtained from the discretization of the continuous variables associated to each detected leaf (growth rate, mean size of leaves, etc) (Sec. 2.2). The same is applicable to the grounding of the symbolic actions, which consist of discrete values of amount of water and nutrient, which can be easily transformed into low-level signals to control the probe mounted on the robot hand to administer the required doses (Sec. 2.3).

Another important problem we faced during the implementation of the cognitive architecture was the noise generated by the sensing mechanisms. In our architecture, the symbolic state description is defined from continuous properties measured by a real camera system with several causes for noisy or erroneous measurements. Consequently, state transitions from one quantized state to the other (e.g. medium leaf to large leaf) are sometimes noisy and unreliable. We deal with this problem using two strategies. The first one consists in selecting an appropriately coarse time interval for sampling

and planning (every 24 hours). In this case, filtering, like a voting mechanism, helps to get more reliable values. The second one is the probabilistic approach to generate cause-effects rules. Note that a noisy observation would explain only one experience from the dataset. Thus, its impact on the calculated probabilities would be very limited. An unreliable perception can also be produced when leaves from different plants occluded each other. To tackle this problem, experiments were stopped before leaves from different plants overlapped. The good results obtained in the experiment in the real scenario (Sec. 3) show that our system is able to handle noisy measurements.

From a software engineering viewpoint, integration into a heterogeneous software system takes some extra effort compared to simulations being run within the same programming environment. We encapsulated different functionalities in ICE modules (Internet Communication Engine) [39] in order to run modules smoothly together on different computers under different operation systems, implemented in different languages.

Another difficulty that we needed to address to make the cognitive architecture viable was the time of response of the human gardener. In the simulated scenario, a new cause-effect rule may be made available any time (time stops when no input comes), whereas a real system waiting for response hangs while the real plants continue evolving. Since a human cannot be available for helping the system 24/7, the time interval for planning was set to once a day, only. This requires the operator/gardener to look after the system only once a day during a predefined short time frame.

Another aspect to stress is that the variables configuring the state space are predefined in advance. Currently, the decision-making approach does not include any mechanism for the automatic generation of these variables. This could affect the performance of the system in situations in which the state space variables do not permit distinguishing between, for instance, a plant having an excess of water or a plant having a disease. In that case, it would be necessary to manually include new variables in the state space definition to disambiguate between these situations. We would like to note that, if a mechanism for generating new state variables during runtime were provided, the system would be able to incorporate these variables without deteriorating much its performance. This is so since the learning mechanisms would start considering the new variables for the refinement of cause-effect rules, while keeping the already existing ones.

Finally, we would like to remark that, even though we selected water and nutrients to be administered by the effectors of the robot to treat plants, adding other effectors for applying different chemical compounds would be a matter of design, enlarging the treatment space, but not changing the complexity of the problem dramatically. Treating plants with different chemical compounds, like fertilizers, pesticides, fungicides, and also water, precisely where needed, is a central task in precision agriculture [40, 41]. Industrially available solutions include tractors, harvesting machines and field robots as well as UAVs for measuring plant and field status and environmental parameters. Rules for spraying chemicals accurately are mostly relying on experience of farmers and plant experts. The system presented and successfully tested here, may help to automate rule generation and optimize treatment planning and replanning also in this societally relevant application field.

5 Conclusions

In this work we presented a robotic cognitive architecture for the control of the growth of plants. The developed system is capable of providing an individualized care of plants by using artificial intelligence (AI) techniques for planning and learning integrated in a robotic platform. To the best of our knowledge, this is the first robotic system using artificial intelligence developed to this end.

For making decisions, we used the decision-making framework in [4]. This framework interleaves a symbolic planner, capable of generating plans for tasks involving weak and delayed correlations between actions and effects, such as controlling the growth of plants, and a learner, which learns during task execution planning operators coding these weakly correlated cause-effects. The AI decision-making mechanisms were integrated with robotic techniques for sensing and acting for the grounding of the symbolic descriptions of states and actions, respectively. Sensing mechanisms consists of leaf segmentation from images obtained by an eye-in-hand camera using a superparamagnetic clustering approach [30, 31] from which the number, color, and size of leaves were derived. This information was used to generate a symbolic state description for the decision-making approach to decide about the treatment (doses of water and nutrients) that should be applied to the plant. These doses were supplied to the corresponding plant using a probe mounted on the hand of a robot arm [3].

The architecture allows a human gardener to speed up learning by providing the treatment to execute when the planner cannot make a decision due to (still) missing operators. The intervention of the gardener is kept to a minimum since plan generation failure takes place mostly at the beginning of the learning and quickly decreases to zero until the human intervention is no longer required, moment at which the system works fully autonomously.

The cognitive architecture proved to work successfully in a real plant scenario. The system was able to learn relevant cause-effect rules to perform an individualized care of plants efficiently. Unexpected effects were correctly handled, e.g. for plant 880573 in day 3 (see Table 7), where interleaving planning and learning prevented the interruption of the task by generating a new plan from the unexpected situation that permitted completing the task.

Acknowledgments

This research was supported by the commission of the European Communities Seventh Framework Programme FP7/2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No 247947 - GARNICS.

References

[1] B. Blackmore, “A systems view of agricultural robots,” *Precision agriculture*, vol. 7, pp. 23–31, 2007.

[2] G. Alenyà, B. Dellen, S. Foix, and C. Torras, "Robotized plant probing: Leaf segmentation utilizing time-of-flight data," *IEEE Robotics and Automation Magazine*, vol. 20, no. 3, pp. 50–59, 2013.

[3] A. Fischbach, "Automatische visuelle posenkorrektur eines roboterarms," Master's thesis, Fachhochschule Aachen, Campus Jülich, 2011.

[4] A. Agostini, C. Torras, and F. Wörgötter, "Learning Weakly Correlated Cause-Effects for Gardening with a Cognitive System," *Engineering Applications of Artificial Intelligence*, vol. 36, no. 0, pp. 178–194, 2014.

[5] GARNICS, "GARdeNIng with a Cognitive System." http://www.garnics.eu/, 2010-2013.

[6] D. Houle, D. Govindaraju, and S. Omholt, "Phenomics: the next challenge," *Nature Review Genetics*, vol. 11, no. 12, pp. 855–866, 2010.

[7] R. Furbank and M. Tester, "Phenomics technologies to relieve the phenotyping bottleneck," *Trends in Plant Science*, vol. 16, no. 12, pp. 635–644, 2011.

[8] H. Scharr, M. Minervini, A. Fischbach, and S. Tsaftaris, "Annotated image datasets of rosette plants." http://juser.fz-juelich.de/record/154525/files/FZJ-2014-03837.pdf, 2014.

[9] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsaftaris, "Finely-grained annotated datasets for image-based plant phenotyping," *Pattern Recognition Letters: Special Issue on Fine-grained Categorization in Ecological Multimedia*, pp. 1 – 11, 2015.

[10] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, 2014.

[11] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[12] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 335–346, 1990.

[13] N. Krüger, J. Piater, C. Geib, R. Petrick, S. M, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrcen, A. Agostini, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems RAS*, vol. 59, no. 10, pp. 740 – 757, 2011.

[14] M. Beetz, L. Mösenlechner, and M. Tenorth, "Crama cognitive robot abstract machine for everyday manipulation in human environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1012–1017, IEEE, 2010.

[15] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: a constraint-based approach," in *Proceedings of robotics: science and systems*, 2016.

[16] C. Paxton, F. Jonathan, M. Kobilarov, and G. D. Hager, "Do what i want, not what i did: Imitation of skills by planning sequences of actions," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3778–3785, IEEE, 2016.

[17] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning Theory and Practice*. Elsevier Science, 2004.

[18] R. Szeliski, *Computer vision: algorithms and applications*. Springer, 2010.

[19] C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, p. 20, 2007.

[20] I. Dianov, K. Ramirez-Amaro, P. Lanillos, E. Dean-Leon, F. Bergner, and G. Cheng, "Extracting general task structures to accelerate the learning of new tasks," in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pp. 802–807, IEEE, 2016.

[21] N. Correll, N. Arechiga, A. Bolger, M. Bollini, B. Charrow, A. Clayton, F. Dominguez, K. Donahue, S. Dyar, L. Johnson, *et al.*, "Indoor robot gardening: design and implementation," *Intelligent Service Robotics*, vol. 3, no. 4, pp. 219–232, 2010.

[22] B. Al-Beeshi, B. Al-Mesbah, S. Al-Dosari, and M. El-Abd, "iplant: The greenhouse robot," in *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*, pp. 1489–1494, IEEE, 2015.

[23] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*, vol. 2, pp. 762–769, IEEE, 2004.

[24] S. LaValle, *Planning algorithms*. Cambridge Univ Pr, 2006.

[25] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, X. Yin, and S. A. Tsaftaris, "Leaf segmentation in plant phenotyping: a collation study," *Machine Vision and Applications*, vol. 27, no. 4, pp. 585–606, 2016.

[26] M. Giuffrida, M. Minervini, and S. Tsaftaris, "Learning to count leaves in rosette plants," in *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)* (H. S. A. Tsaftaris and T. Pridmore, eds.), pp. 1.1–1.13, BMVA Press, 9 2015.

[27] B. Romera-Paredes and P. H. S. Torr, "Recurrent instance segmentation," *CoRR*, vol. abs/1511.08250, pp. 1–23, 2015.

[28] M. Ren and R. S. Zemel, "End-to-end instance segmentation and counting with recurrent attention," *CoRR*, vol. abs/1605.09410, pp. 1–14, 2016.

[29] B. Dellen, H. Scharr, and C. Torras, "Growth signatures of rosette plants from time-lapse video," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. PP, no. 99, pp. 1 – 11, 2015.

[30] A. Abramov, K. Pauwels, J. Papon, F. Wörgötter, and B. Dellen, "Real-time segmentation of stereo videos on a portable system with a mobile gpu," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 22, pp. 1292–1305, 2012.

[31] E. E. Aksoy, A. Abramov, F. Wörgötter, H. Scharr, A. Fischbach, and B. Dellen, "Modeling leaf growth of rosette plants using infrared stereo image sequences," *Computers and Electronics in Agriculture*, vol. 110, pp. 78 – 90, 2015.

[32] E. Finkel, "With phenomics plant scientists hope to shift breeding into overdrive," *Science*, vol. 325, pp. 380–381, 2009.

[33] K. Nagel, A. Putz, F. Gilmer, K. Heinz, A. Fischbach, J. Pfeifer, M. Faget, S. Blossfeld, M. Ernst, C. Dimaki, B. Kastenholz, A.-K. Kleinert, A. Galinski, H. Scharr, F. Fiorani, and U. Schurr, "GROWSCREEN-Rhizo is a novel phenotyping robot enabling simultaneous measurements of root and shoot growth for plants grown in soil-filled rhizotrons," *Functional Plant Biology*, vol. 39, pp. 891–904, 2012.

[34] M. Minervini, H.Scharr, and S. Tsaftaris, "Image analysis: the new bottleneck in plant phenotyping," *Signal Processing Magazine*, vol. 32, pp. 1 – 6, July 2015.

[35] A. Agostini, C. Torras, and F. Wörgötter, "Efficient Interactive Decision-making Framework for Robotic Applications," *Artificial Intelligence*, 2015. http://dx.doi.org/10.1016/j.artint.2015.04.004.

[36] R. Petrick and F. Bacchus, "A knowledge-based approach to planning with incomplete information and sensing," in *AIPS*, pp. 212–221, 2002.

[37] X. Wang, "Learning by observation and practice: An incremental approach for planning operator acquisition," in *ICML*, pp. 549–557, 1995.

[38] T. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, 1997.

[39] http://zeroc.com/products/ice.

[40] J. V. Stafford, "Implementing precision agriculture in the 21st century," *Journal of Agricultural Engineering Research*, vol. 76, no. 3, pp. 267–275, 2000.

[41] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, 2012.