Auton Robot (2015) 39:469–485
DOI 10.1007/s10514-015-9471-y

Cognitive robots learning failure contexts through real-world experimentation

Sertac Karapinar[1] · **Sanem Sariel**[1]

Received: 11 December 2014 / Accepted: 9 July 2015 / Published online: 7 September 2015

Abstract Learning is essential for cognitive robots as humans to gain experience and to adapt to the real world. We propose an experiential learning method for robots to build their experience online and to transfer knowledge among appropriate contexts. Experience gained through learning is used as a guide to future decisions of the robot for both efficiency and robustness. We use Inductive Logic Programming (ILP) learning paradigm to frame hypotheses represented in first-order logic that are useful for further reasoning and planning processes. Furthermore, incorporation of background knowledge is also possible to generalize the framed hypotheses. Partially specified world states can also be easily represented by these hypotheses. All these advantages of ILP make this approach superior to the other supervised learning methods. We have analyzed the performance of the learning method on our autonomous mobile robot and on our robot arm both building their experience on action executions online. It has been observed in both domains that our experience-based learning and learning-based guidance methods frame sound hypotheses that are useful for constraining and guiding the future tasks of the robots. This learning paradigm is promising especially for the contexts where abstraction is useful for efficient transfer of knowledge.

Keywords Robot learning · Learning from experience · Experiential learning in robots · Planning and plan execution · Action execution monitoring · Real-world experimentation · Failure contexts

✉ Sanem Sariel
sariel@itu.edu.tr

Sertac Karapinar
karapinarse@itu.edu.tr

[1] Artificial Intelligence and Robotics Laboratory, Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey

1 Introduction

Robot skills are acquired and improved through learning. Several earlier studies present different methods for robots to learn to complete tasks more efficiently; and task completion is always the highest priority in these studies. However, especially in unstructured environments, there are cases where task completion is not possible, or certain precautions should be taken into account to ensure safety in task execution. In this research, we study how learning helps a robot determine general or specific limitations on task execution beyond its capabilities, and gain experience on these cases to make failure/risk-averse decisions on future tasks. Thus, instead of learning something novel or to do something better, the robot learns in which circumstances it is better not to do something but to consider alternative options.

Our particular focus is on the problem of learning from real-world observations of persistent failures or undesired effects without prior models of them, and automatic derivation of hypotheses that relate execution contexts to failure cases in an experiential learning process (Kolb 1984). These hypotheses are further used by robots to improve task execution efficiency in constrained cases, to ensure robustness or to avoid any potential damages to their surrounding worlds or to themselves.

A motivating example can be given to illustrate the learning problem in the blocks world environment where a humanoid robot stacks blocks to build a given structure. The robot may discover that particular objects are hard to be localized by its existing onboard vision system. In a time-constrained case, instead of targeting those blocks, it may

decide to build the construction with blocks that can be localized more quickly. Consider another case in which action *stack* fails when the height of the stacked objects gets taller. In such a scenario, the learner needs to relate the failure with the height of the block stack that is in interest through observation of several similar situations. This experience further can be extended for stacks of different types of objects. Our work builds on our previous work suggesting such reasoning mechanisms (Usug and Sariel-Talay 2011; Usug et al. 2012; Karapinar et al. 2012; Karapinar and Sariel-Talay 2012), and proposes real-world experience-based learning and learning-based guidance methods for constrained decision making in robots (Karapinar et al. 2013; Yildiz et al. 2013; Karapinar and Sariel 2015; Sariel et al. 2015).

Contributions of this paper are threefold. (1) We propose to use a knowledge-based inductive learning approach that can extract contextual information for building experience. We present an experiential learning framework based on Inductive Logic Programming (ILP) for robots to build and continually maintain experience on potential risks, limitations and constraints on task execution without prior models. Hypotheses that relate execution contexts to action outcomes are derived using the learning process based on online observations made in the real world. Symbolic predicates on actions, relevant features of objects and their spatial relations are encoded in contexts of hypotheses representing failure cases in First Order Logic (FOL) sentences, and the learning process can use background knowledge whenever it is available. We demonstrate that by using background knowledge on the derived hypotheses, more realistic unifications, abstractions or generalizations could be extracted which are useful for inter-domain knowledge transfer. (2) We present automated planning guidance methods that use the gained experience in further decisions of robots by imposing domain constraints using the derived hypotheses. This strategy fits in an adaptive planning framework to constrain decisions based on experience. (3) We validate our methods on two autonomous robot platforms executing object manipulation tasks without any human intervention. These robots are exposed to the challenges of unstructured real-world environments including noisy sensory data, changes in illumination for object recognition and unexpected outcomes of actions. We show that without a model-based failure isolation, robots can determine their limitations by experience-based learning, and ensure robustness and safety by learning-guided planning. Therefore, both failure handling and prevention is ensured to a great extent.

Throughout the paper, we first present earlier works on experience-based learning and learning-guided planning for constrained decision making. Then, we describe the details of our learning and adaptive planning methods. We analyze the performance of our methods on two different domains with our autonomous robots, and present the results of these analyses in the experimental results section. Then, we put together all the findings, and present the promising features of our methods and the future directions in the discussion section. Finally, we conclude the paper.

2 Related work

Learning can be used either as a model construction tool for affordances or action schema (Pasula et al. 2007; Lang and Toussaint 2010; Hermans 2014; Magnenat et al. 2012) or for improving performances of robots in dealing with challenges arising from inconsistencies of prior models (Gspandl et al. 2011) during execution of actions in the real world. One of the most important works in the area addresses learning stochastic models of action schemas (Pasula et al. 2007). In their work, probabilistic STRIPS rules for actions are learned from offline training data. Lang and Toussaint (2010) further extend this work to use learned schemas in probabilistic planning. In a recent work (Hermans 2014), learning is used to predict the effects of affordances. Visual attributes of objects are used for affordance prediction. Dearden and Burbridge (2014) present a supervised learning approach for mapping between geometric states and logical predicates in robot manipulation tasks.

One of the objectives of this work is to develop methods to use the gained experience to ensure robustness and safety. One appropriate way is incorporating the experience into the search process during symbolic planning. In the context of planning, usually heuristics or control knowledge are used to reduce the search complexity. Alternative heuristic methods exist for macro-operators, search control rules or policies (Bacchus and Kabanza 2000; Botea et al. 2005; López and Plaza 1991; Khardon 1997; Wang et al. 1996) which can be automatically derived (Aler et al. 2002; Borrajo and Veloso 1994, 1996; Cohen 1990; Estlin and Mooney 1997; Fernandez et al. 2004; Leckie and Zukerman 1998). In some of these works, previously generated plans with hand-coded rules are used as training examples for learning. In most of these approaches, control-rules are learned from planning failures (Katukam and Kambhampati 1994) to speed-up planning (Duran 2006; Rintanen 2000).

Our strategy in incorporating experience into planning is using incrementally derived hypotheses for constraining robots' decisions in new planning tasks. One of the most similar studies to our strategy (Haigh and Veloso 1999) integrates planning, monitoring and learning processes to estimate action costs and probabilities from real-world observations and creates control rules to avoid failures. In this system, a regression learning approach is used to find relations among features of the environment, action costs and situations. Derived control rules are used to make

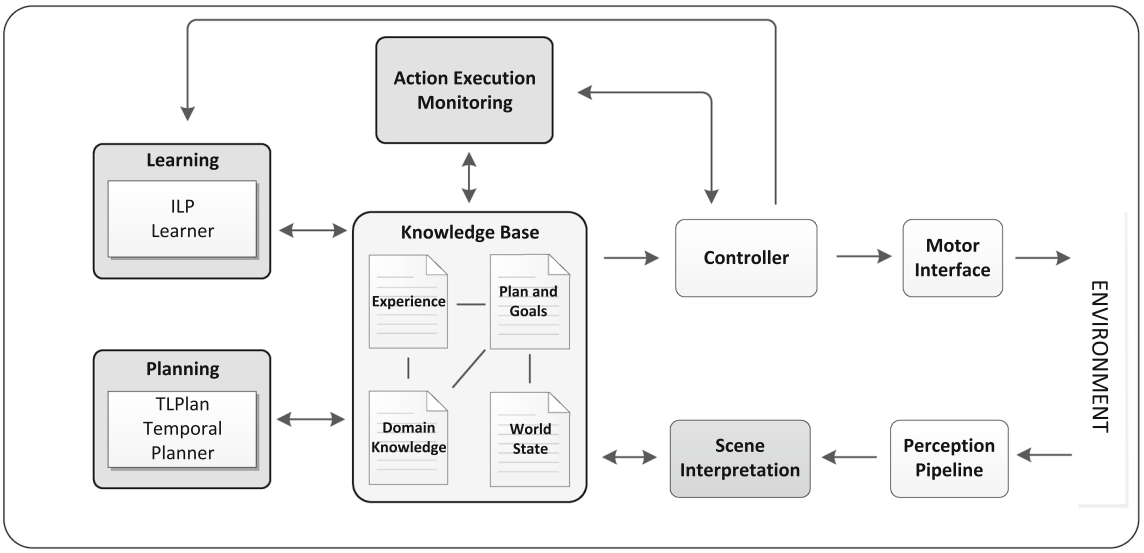


Fig. 1 The overall learning framework for cognitive robots. The framework includes *Perception Pipeline*, *Scene Interpretation*, *Planning*, *Learning*, *Controller*, *Action Execution Monitoring* and *Motor Interface* components which are tightly connected to each other and run in parallel

decisions on goal and action selection. In another study (Morisset and Ghallab 2008), robust ways of performing tasks are learned. These studies address navigation tasks. Learning outcomes represent path-costs depending on different situations.

In some studies, real-world constraints are investigated for decision making in robots. Aoude et al. (2010) propose methods for treat-aware path planning that consider intentions of other drivers. Feyzabadi and Carpin (2014) investigate risk-awareness in path planning by constrained Markov Decision Processes. In another work (Stansbury and Agah 2012), robot decisions are made by constraint programming to ensure safety. Similarly Chow et al. (2014) analyze safety criteria in multi robot decision making for robustness. We extend these ideas to address domains where symbolic knowledge can be useful in building experience by learning from experimentation. Rather than only using an offline learning process, we focus on methods that ensure experimentation of robots in the real world. We particularly study cases for ensuring safety. Our research question is that when a robot discovers online that its actions have some potential damages to the environment how it should represent these cases if it does not have the model of the underlying reasons, and how it can prevent these cases from recurring again. So we want the robot incrementally learn and build its experience from its online experimentation, then, impose correct constraints in its planning domain based on its experience. Therefore, the conclusions made by the robot should be as generic as possible for correct reasoning and knowledge transfer across domains. Knowledge-based learning methods are needed for symbolic reasoning and making abstractions on conclusions which can be used to guide new planning tasks.

3 Experience-based learning of failure contexts

We study the *real-world experience-based learning problem* in which a robot is expected to learn from its observations online. In the first part of the problem, given a set of observations on outcomes of executions in the real world, the task is to determine patterns on failure cases. This is needed to hypothesize such cases without underlying models. A knowledge-based learning approach is more suitable for generalization of these hypotheses. The second part of the problem focuses on preventing these failure cases recurring again. So, given the experience built by the robot, the second task is to make use of this experience for constraining decisions of the robot in achieving new goals.

We propose a real-world experimentation and continual learning framework (Karapinar et al. 2012) to meet the requirements of the *real-world experience-based learning problem* for a cognitive robot. The framework includes *Perception Pipeline*, *Scene Interpretation*, *Planning*, *Learning*, *Controller*, *Action Execution Monitoring* and *Motor Interface* components (Fig. 1). *Perception Pipeline* collects data from different sensor modalities, and the processed data are filtered by *Scene Interpretation* to interpret the world state. The *Planning* component is responsible for generating a safe plan as a sequence of domain actions ($a_i \in A$) given the up-to-date domain knowledge, the goals and the experience. In our case study, an automated planner, TLPlan (Bacchus and Kabanza 2000) is used but if needed, these plans may be hand-coded as well. *Controller* runs the required behaviors to execute a symbolically constructed plan in the real world. *Action Execution Monitoring* continually monitors execution of each action a_i and keeps track of observations related to

outcomes of execution. The *Learning* component continually updates the experience on failure cases. All of the components are designed to be tightly connected to each other, to run in parallel and to access to and maintain a Knowledge Base (KB) consisting all the domain knowledge, the world state observation history and the experience gained online. The current implementation is made available under the robot operating system (ROS) framework (Quigley et al. 2009). Note that only the high-level components are shown in the figure, and low-level components such as localization, mapping, path planning are not illustrated.

We first present some important definitions used in our formulation for solving the investigated problems.

Definition Observed outcome of action a_i is represented with the following triple (a_i, t, out_t) where $out_t \in \{success, failure\}$ and t is the corresponding time step. Outcome $success$ is determined at the end of the execution of a_i while outcome $failure$ is detected at time step t during the execution of a_i by *Action Execution Monitoring*.

Definition An observation context (c_t) is registered for each observed outcome. It is composed of a conjunction of predicates representing either predetermined or observed features (e.g., $category$, $shape$, $color$, $size$, $location$, etc.) of objects to be manipulated, spatial relations among objects (e.g., $on(obj_1, obj_2)$ representing that the first object is on top of the second object), the internal state of the robot (e.g., $holding(obj_3)$) and the known or the other perceivable features of the world state (e.g., $soundDetected$ indicating that a change in the environment's sound is detected by the robot's microphone). Note that it is not always possible to collect correct information on all true facts due to limitations in sensing. So, observation contexts may also be noisy.

Definition An observation obs_t made at time step t combines an outcome (a_i, t, out_t) for a_i with its corresponding execution context c_t. The robot continually builds its real-world observation history Obs formed by each obs_t, $0 \leq t \leq T$.

For each a_i, sets of positive examples ($P(a_i)$) and negative examples ($N(a_i)$) can be defined as follows:

$$P(a_i) = \bigcup_{(a_i, t, out_t = failure)} obs_t \quad (1)$$

$$N(a_i) = \bigcup_{(a_i, t, out_t = success)} obs_t \quad (2)$$

where positive examples correspond to failure cases while negative examples to success cases. This is because that modelling failure cases is the main objective.

Definition Background theory $\mathcal{B}$ is represented as a conjunction of rules in Horn Clause (Horn 1951) representing prior information and commonsense knowledge. For example, $in(X, Y) \wedge in(Y, Z) \Rightarrow in(X, Z)$ represents the transitive property of relation in for forward chaining reasoning on nested objects as substitutions of X, Y and Z.

Given these definitions, the objective of the *real-world experience-based learning* process is to continually maintain a hypothesis space $\mathcal{H}$ as a disjunction of hypotheses for each a_i, that explains $P(a_i)$ and rejects $N(a_i)$ by applying reasoning steps to generalize each $h_j \in \mathcal{H}$ using $\mathcal{B}$. Each h_j represents an implication with its premise as a failure context and its conclusion as an observed action outcome. Whenever needed, new hypotheses are framed based on new incoming observations, or existing ones are ruled out, and the KB is updated according to this inference result. We use Inductive Logic Programming paradigm for real-world experience-based learning.

In our approach, derived hypotheses framed from observations are used to provide feedback to the robot to improve its performance on its future tasks which enables real-world experimentation (Karapinar et al. 2013; Karapinar and Sariel 2015). An adaptive planning strategy uses these hypotheses to guide future planning processes of the robot (Yildiz et al. 2013; Karapinar et al. 2013; Sariel et al. 2015).

We focus on ground and tabletop object manipulation scenarios in which our robots localize the existing objects in their environments, and execute actions from their action sets ($A_{ground} = \{move, pickUp, transport, putDown\}$ for ground manipulation scenarios and $A_{tabletop} = \{move, pickUp, transport, stack\}$ for tabletop scenarios). We place special emphasis on *Scene Interpretation* and *Action Execution Monitoring* components due to their important roles in the overall success of the real-world experience-based learning framework. We summarize the processes running in these components here for convenience.

3.1 Scene interpretation

The raw visual and non-visual sensory data acquired by the sensors of the robot are processed through a series of different operations running in parallel, and the processed data are presented to *Scene Interpreter* which encodes and updates its KB (Ozturk et al. 2014; Ersen et al. 2014). The robot uses its laser rangefinders and encoders for *Localization and Mapping*. *Scene Interpreter* uses visual perceptual data taken from the RGB-D sensors for 3D object recognition (Hinterstoisser et al. 2012) and Euclidian clustering-based segmentation (Trevor et al. 2013). When the robot interacts with known objects, it can use their predefined models (i.e., templates) and different physical/visual features (Ersen et al. 2013). If the object models are not known in advance, the robot is expected to extract some of the observable visual features (e.g., shape, color and size) of objects and their relations

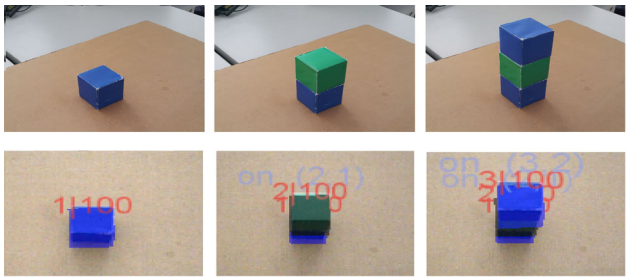


Fig. 2 (*top*) Real-world 3-block stacking snaphots (*bottom*) The representations of these blocks and the *on* relations among them from the viewpoint of the robot. The blocks are labelled as 1, 2 and 3 from *bottom* to *top*, respectively, and the confidence of the existence of each object is also shown along with the object id

(Ersen et al. 2014). In object manipulation tasks, extracting spatial relations among the objects is of great importance as well as the attributes describing these objects. The following spatial predicates are computed based on the locations of and the relations among objects:*on*, *near*, *on_table*, *on_ground* and *clear* (Ozturk et al. 2014; Ersen et al. 2014). *Scene Interpreter* incorporates and filters all these data temporally to build the knowledge on objects along with their attributes, the estimated locations in the global map and the spatial relations among them. It has been shown that this component can handle sensor noise, changes in illumination, occasional localization failures and dynamic situations. A sample interpretation for a 3-block tabletop stacking scenario is given in Fig. 2. When an object is first recognized in the scene, a new id is assigned, and its location is registered along with the type and color information. Furthermore, relations among objects are also detected during runtime. The blocks are labelled as 1, 2 and 3 from bottom to top, respectively. In this example, $on(2,1)$ and $on(3,2)$ relations exist. The colors of blue objects are recognized correctly while that of block 2 is not. In this case, color information of this block is registered as unknown to be updated whenever it can be detected. Although not shown in the figure for clarity, $on\_table(1)$ and $clear(3)$ predicates are determined. Note that $on\_table(1)$ predicate is detected in the first frame, and remains in the KB during execution although block 1 is occluded by the objects on top of them. The same is true for all other information regarding to occluded objects.

Non-visual data are also extracted from different sensor modalities including sonar sensors, microphones and pressure sensors in the end effector of the robot. All incoming data are fused, and corresponding symbolic predicates are maintained in the KB without any apriori information on object locations in the environment. These predicates are created and updated over time by *Scene Interpreter* to maintain the consistency of the world model during runtime. Later, these predicates are to be used for monitoring execution.

3.2 Action execution monitoring

Monitoring is needed to continuously track the execution of a given plan to check the updated states and detect failures if any. If an observed world state does not include the intended outcome of an action, a failure is assumed. We use a model-based monitoring system (Kapotoglu et al. 2014) which is suitable for detecting failures either during or after execution of actions. In this system, Metric Temporal Logic (MTL) is used in order to represent action monitoring formulas for failure detection (Kvarnström et al. 2008). MTL is a logic-based language which includes temporal logic operators as well as standard boolean connectives. A goal progression algorithm (Kabanza et al. 1997) is used to evaluate these MTL formulas. For each action, a set of action monitoring formulas are intuitively defined. The start and finish times of an action in execution and several predicates about task execution are monitored during execution. When the robot starts executing an action, the action monitoring formulas are added to the control formula. If a failure is detected, the robot ceases the execution for recovery. The observations made in failure cases are also encoded in the KB for learning from experience.

3.3 Learning

In this work, we use and enhance the Progol algorithm (Muggleton 1995) to apply inductive logic programming (ILP) paradigm for *real-world experience-based learning*.

3.3.1 The Progol algorithm

We adopted the Progol algorithm (Muggleton 1995) based on the constraints of the experience-based learning problem. The Progol algorithm uses the mode directed inverse entailment (MDIE) approach. It benefits from mode declarations for describing the predicates in a clause. The following mode is a body declaration stating that predicate $category$ takes two arguments.

$$:-modeb(1, category(+object, \#categoryType))? \quad (3)$$

where 1 indicates a binary value ($true/false$) for predicate $category$. The first argument is a variable of type $object$, and the second one is a constant value. An example use of this declaration is $category(obj_1, box)$. The other predicates are encoded in the same way. The possible outcomes ($success, failure$) are taken as the classes for binary classification.

In general, *real-world experience-based learning problem* can be formulated in the ILP framework as follows. Given Background theory $\mathcal{B}$ and observations Obs, finding the related hypothesis space $\mathcal{H}$ such that

$$\mathcal{B} \wedge \mathcal{H} \models Obs \tag{4}$$

By using contraposition law:

$$\mathcal{B} \wedge \overline{Obs} \models \overline{\mathcal{H}} \tag{5}$$

where $\overline{\mathcal{H}}$ and $\overline{Obs}$ are ground skolemised unit clauses if $\mathcal{H}$, and Obs are Horn clauses (Muggleton 1995). If we define $\overline{\perp}$ as the conjunction of true ground literals in every models of $\mathcal{B} \wedge \overline{Obs}$, this implies:

$$\mathcal{B} \wedge \overline{Obs} \models \overline{\perp} \tag{6}$$

As we know that $\overline{\mathcal{H}}$ must be true in every model of $\mathcal{B} \wedge \overline{Obs}$, it includes a subset of the $\overline{\perp}$.

$$\mathcal{B} \wedge \overline{Obs} \models \overline{\perp} \models \overline{\mathcal{H}} \tag{7}$$

which implies Muggleton (1995):

$$\mathcal{H} \models \perp \tag{8}$$

By using this rule, the algorithm finds a subset of $\mathcal{H}$ theta-subsuming $\perp$ instead of searching for a large set of candidates. In fact, this process defines two important steps of the general algorithm: generating the most specific clause ($\perp$) and searching the subsumption lattice. The main objective of the algorithm is to frame this most general hypothesis space explaining positive examples. To see how the most specific clause is built, assume that the robot has the following background knowledge and the observation:

B: $shape(X, prism) \Rightarrow category(X, box)$,
$category(X, box) \Rightarrow size(X, large)$
obs_1: $category(obj_1, box) \wedge outcome(pickUp(obj_1), failure)$

$\perp$ is constructed as:

$\perp$: $category(obj_1, box) \wedge shape(obj_1, prism) \wedge$
$size(obj_1, large) \Rightarrow outcome(pickUp(obj_1), failure)$

The main process is presented in Algorithm 1 which can be described in four steps. First, a positive example to be generalised is selected. If there is not any, the algorithm stops. The observation history of the robot is kept in the Obs array (line 2). For each observation in the array, lines 3–9 are executed. In the second step, the most specific clause entailing the selected example is generated (line 4). The selected clause includes the subset of the predicates of the positive example.

In the next step, a hypothesis h_i is generated using the most specific clause (line 5). The search algorithm is given in Algorithm 2.

Hypothesis search is bounded by the empty clause and the most specific clause ($\square \leq \mathcal{H} \leq \perp$). As can be seen, the most specific clause has a significant effect on the search space. The search is from general to specific, and each candidate clause is tested against how many positive and negative examples are predicted. Progol uses an A*-like algorithm to find the subsumption lattice. Search space is a tree such that its root node is $\square$, and a parent node is a theta-subsumption of its children nodes. In Algorithm 2, $Open$ and $Closed$ lists keep the clauses to be evaluated (nodes in the tree) and hypothesis candidates respectively. From the $Open$ list, the best clause (s) is selected (line 3 calls Algorithm 3) and is added to the $Closed$ list (line 4). The successor clauses are added to the $Open$ list, and the candidate clauses are removed from the $Open$ list (line 5). Algorithm 2 terminates when the $Open$ list includes the most specific clause ($\perp \subset Open$). It returns the candidate clause with the best score as a hypothesis (line 8). The score of a clause is calculated by using the number of positive and negative examples entailed by that clause, $f_s = p_s - n_s$ (Algorithm 3). f_s is a measure of how well the clause explains the observations.

In the final step of Algorithm 1, the clause with the best score is added to the hypothesis space, positive examples explained by this hypothesis are removed (line 6–8). Then, the algorithm returns to line 3.

We enhanced this algorithm to interpret ambiguous observations, and we provided that probability values are also attached to hypotheses. To assign probability value to a hypothesis, we use the ratio of positive and negative obser-

Algorithm 1: Progol-Learning(a_i, KB)

input : action a_i, KB to get observations Obs
output: hypothesis space, $\mathcal{H}$

1 $\mathcal{H} = \varnothing$
2 Obs=KB.getObservationHistory()
3 **foreach** $obs_t \in P(a_i)$ **do**
4 $\quad \perp$ = MostSpecificClause(obs_t)
5 $\quad h_i$ = Search($\perp$)
6 $\quad \mathcal{H} = \mathcal{H} \cup h_i$
7 $\quad Obs' = \{obs : obs \in Obs$ and $\mathcal{H} \models obs\}$
8 $\quad Obs = Obs \setminus Obs'$
9 **end**

Algorithm 2: Search($\perp$)

input : the most specific clause, $\perp$
output: a more general clause if exists

1 $Open = \{\square\}$, $Closed = \varnothing$
2 **while** $\perp \notin Open$ **do**
3 $\quad s = Best(Open)$
4 $\quad Open = Open \setminus s$, $Closed = Closed \cup s$
5 $\quad Open = Open \cup MoreSpecificClauses(\perp, s) \setminus Closed$
6 **end**
7 $Closed = Closed \cup Best(Open)$
8 return $Best(Closed)$

Algorithm 3: Best(S)

input : a set of clauses S
output: a clause with the best score, s^*

$f^* = -\infty, s^* = \varnothing$
foreach $s \in S$ **do**
  $f_s = p_s - n_s$
  if $f_s > f^*$ *and* $p_s + n_s \geq 5$ **then**
    $s^* = s$
  end
end
return s^*

vations covered by that hypothesis: $P = p_s/(p_s + n_s)$. We have also seen in our preliminary results that the probability values make more sense after considering a sufficient number of observations. For this purpose, $p_s + n_s \geq 5$ condition is added to Algorithm 3.

Algorithm 2 takes into account of the permutations of the attributes that might take place in hypotheses. Therefore, it has the complexity of $O(2^d)$ where d is the number of the attributes. Overall, the main algorithm has a complexity of $O(n.2^d)$ where n is the number of positive examples.

3.3.2 Hypothesis framing for failure cases

To illustrate how hypotheses are framed for failure cases, a ground robot scenario can be given as follows. Assume that the following observations are taken by the robot:

$$obs_1 : category(obj_1, box) \wedge shape(obj_1, prism) \wedge color(obj_1, green) \wedge material(obj_1, paper) \wedge size(obj_1, small) \wedge outcome(pickUp(obj_1), success)$$

$$obs_2 : category(obj_2, box) \wedge shape(obj_2, prism) \wedge color(obj_2, black) \wedge material(obj_2, paper) \wedge size(obj_2, large) \wedge outcome(pickUp(obj_2), success)$$

$$obs_3 : category(obj_3, box) \wedge shape(obj_3, cylinder) \wedge color(obj_3, red) \wedge material(obj_3, paper) \wedge size(obj_3, large) \wedge outcome(pickUp(obj_3), failure)$$

Each observation obs_t corresponds to an instance of an execution of action $pickUp$. A context c_t, in this example represented by the attributes of the object to be picked up is mapped to the outcome of the observed action ($success/failure$). When the ILP learning is applied on this set, the following hypotheses are derived:

$$h_1 : shape(X, prism) \Rightarrow outcome(pickUp(X), success)$$

$$h_2 : shape(X, cylinder) \Rightarrow outcome(pickUp(X), failure)$$

After framing these hypotheses, suppose that the robot gets a new observation:

$$obs_4 : category(obj_4, box) \wedge shape(obj_4, prism) \wedge color(obj_4, green) \wedge material(obj_4, plastic) \wedge size(obj_4, large) \wedge outcome(pickUp(obj_4), failure)$$

After getting this observation, the hypotheses in the KB are generalized as follows:

$$h_1 : shape(X, prism) \wedge material(X, paper) \Rightarrow outcome(pickUp(X), success)$$
$$h_2 : material(plastic) \vee shape(cylinder) \Rightarrow outcome(pickUp(X), failure)$$

Note that the probabilities for these hypotheses are computed as 1 since there are no conflicting observations. In this particular example, only the attributes of a single object are taken into account. However, in a more complex scenario involving many objects, the attributes of all objects and their relations are taken into account.

As in the other supervised learning algorithms, a sufficient number of observations are needed to come up with correct conclusions. The main superiority of the *ILP learner* to the conventional attribute-based classifiers is its knowledge-based representation. It can represent hypotheses in FOL and incorporate background knowledge.

3.4 Mapping from hypotheses to heuristics

The learning procedure continually frames new hypotheses during execution. These hypotheses are to be used to constrain the decisions of the robot on its future tasks. We analyze three ways of using hypotheses: (i) deriving new control formulas (ii) updating the models of operators corresponding to the failed actions (iii) setting an adaptive cost computation method for the corresponding operators. In the first approach, the selection of a failed operator is completely abandoned to prevent its selection on specific contexts. In the second approach, the preconditions of the failed operators are updated to prevent their selection in specific branches during search. In the third approach, the cost values of failed operators are updated to set preference models (Yildiz et al. 2013; Sariel et al. 2015).

Three guidance approaches are analyzed in a scenario with several objects to be picked up and moved to a destination (Yildiz et al. 2013; Sariel et al. 2015). The objects have several observable features some of which are predefined. Assume that the following is a hypothesis framed after observations taken from this environment.

$$category(obj_1, box) \wedge color(obj_1, red) \Rightarrow outcome(pickUp(obj_1), failure)$$

Based on this hypothesis, a search control formula can be constructed in LTL according to the context of the given hypothesis as following:

$$\Box(\forall[X : object(X)](category(X, box)) \wedge (color(X, red)) \implies \bigcirc(\neg holding(X)))$$

where $\Box$ is for universal quantification, making this formula $true$ for all world states, and $\bigcirc$ (next) specifies that the given

formula will be true in the next world state. The whole formula specifies that if the given context is satisfied by the world state (including the features of the objects), the robot believes that it will fail in the execution of action $pickUp$, and the effects of the corresponding operator will not appear in the next world state. The formula represents a reject rule for a failed operator by inserting the distinctive effects of the operator (e.g., $holding$ for $pickUp$ operator) in its conclusion part.

The precondition update for the given hypothesis can be done in the following way:

```
(def-defined-predicate (pickupFailContext ?obj)
  (and
    (= (category ?obj) box)
    (= (color ?obj) red)
   )
)
(def-adl-operator (pickup ?robot ?obj)
  (pre
    (and
      (not (pickupFailContext ?obj))
      (handempty ?robot)
      (....)
    )
  )
  (del (...))
  (add (...))
)
```

This precondition update prevents the selection of operator $pickUp$ for the context defined in the hypothesis representing a failure case.

The cost update method also considers the context of the failure case and increases the cost of the operator by a factor to prevent its selection in a future plan according to the following equation.

$$cost' = cost + P(\mathcal{H}_{pickUp}) * k$$

where the update factor is proportional to the probability of the corresponding hypothesis ($P(\mathcal{H}_{pickUp})$). The gain value, k, is set to a number to guarantee that the cost penalty is greater than the maximum cost value of any other operator.

The effects of these three methods are analyzed in another work Yildiz et al. (2013). According to this analysis, a hybrid approach is adopted which uses precondition update to prevent the selection of a failed operator when the outcome is classified as `fail-unsafe` and cost update to ensure its selection when the failure is classified as `fail-safe`, and there is no alternative action.

In this particular work, our focus is on analyzing the performace of the knowledge-based learning algorithm. For this reason, we investigate two outcome cases `success` and `failure` (addressing `fail-unsafe` states that should be avoided) and use precondition update when a failure occurs. If the state corresponds to a `success`, no change is needed in the planning domain. But a persistent failure case, after a certain number of observations, is indeed taken into account to prevent damages to the environment or to the robot. The premise part (context) of a learned hypothesis is used to encode new preconditions for the planning operator for safe execution.

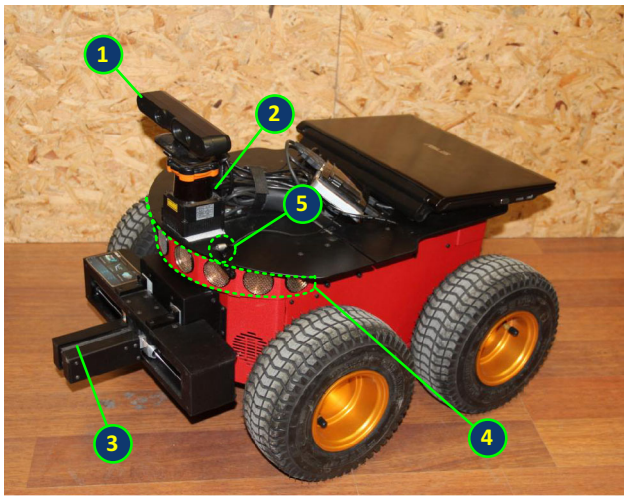
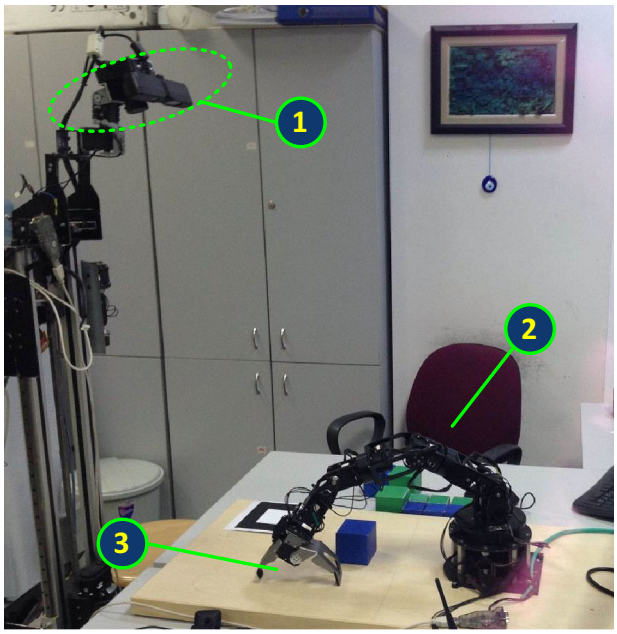


Fig. 3 Gezgin C. with its sensors and effectors

4 Experiments

To evaluate the performance of our methods, we have conducted both ground and tabletop experiments. In the ground experiments, we have used our Pioneer 3-AT mobile robot, Gezgin C. (Fig. 3) equipped with several sensors to sense its environment and detect anomalies. We mounted a Hokuyo UTM-30LX laser rangefinder on top of the robot facing forward for mapping and localization and an ASUS Xtion PRO RGB-D camera on top of the laser rangefinder for 3D object recognition and segmentation. The robot has a 2-DOF gripper to manipulate objects. The gripper paddles have tactile sensors to detect the existence of an object in the gripper.

Tabletop experiments have been done on our 7-DOF Cyton Veta robot arm placed on a table. This robot is connected to a head mounted RGB-D camera system to interpret its scene during runtime. The arm (Fig. 4) with its dexterous design, enhanced motor set to increase its payload and its extension with a Robotis FR07-G101GM gripper set (15 cm gripping size) can grasp several objects placed on the table. The motors placed on the robot arm are connected with a daisy chain structure using RS 485 communication protocol. Beside from the motor commands, the feedback (e.g., position, position error, velocity, load and temperature) from motors are communicated through this communication link controlled by a USB2Dynamixel module.



① ASUS Xtion PRO RGB-D Camera Head
② 7-DOF Cyton Veta Arm ③ Robotis FR07-G101GM gripper

Fig. 4 7-DOF Cyton Veta Robot arm and the head camera system

The processes of both robots are designed to be running on ROS and are compatible with existing low-level components (e.g., localization, mapping, trajectory planning, etc.).

4.1 Overall ground object manipulation performance

We investigate how Gezgin C. autonomously learns in a task of cleaning the environment by manipulation of objects randomly scattered around. In this task, the robot starts its plan by first finding the locations of the objects in the environment. If it cannot detect or recognize any object in its current field of view, it executes action *search* to be able to explore around. In action *search*, the robot starts turning until detecting and/or recognizing an object. Whenever it finds an object, it stops its movement and executes $move - pickUp - transport - putDown$ actions in sequence, if there is no failure. Whenever a failure is detected, the corresponding observation along with its observation context is encoded in the KB as well as the successful completions. Actions are monitored according to the rules governed by *Action Execution Monitoring*. All processes run online on the robot without any supervision. Supplementary sensing actions are also designed for detecting failures occurring out of sensor range. For example, for action $pickUp$, when the object is out of view (when it is close to the robot), the RGB-D sensor cannot detect it. If the robot senses that there is a failure from the tactile sensors, it moves backward to decide if the object is in its original form, and another trial would be safe. If the robot can not sense the object, an unknown failure is assumed, and the observation context with outcome $failure$ is registered to the KB. In the same manner, for action $putDown$, there is no way to check whether the outcome of the action is $success$ without executing sensing actions because the robot releases the object in this case.

Experiments were conducted in our department's corridor without any special lighting. We also let people watch the experiments during which their movements change illumination conditions. The robot maintains its KB during runtime without any human intervention in the face of the challenges of noise in sensory data, partial observability and unexpected situations. The robot interacts with ten objects with different attributes: green and red cylindrical boxes, green, red and blue plastic bowling pins, purple, yellow and pink small balls, a big orange ball and a big beach ball. The initial configurations of these objects and their interpretations in the world model of the robot are visualized in Rviz[1] and presented in Fig. 5. Note that the farther objects are not recognized in the start state. The robot encodes its observations to its KB so that it can build its hypotheses. A sample observation for the ground robot is given below:

$$category(obj_1, pin) \wedge material(obj_1, plastic) \wedge deformability(obj_1, solid) \wedge color(obj_1, blue) \wedge width(obj_1, narrow) \wedge height(obj_1, medium) \wedge locX(obj_1, 0.58) \wedge locY(obj_1, -0.40) \wedge outcome(pickUp(obj_1), failure)$$

Note that the x and y locations ($locX$ and $locY$) represent the positions with respect to the global map.

We first report the overall performance of mobile manipulation by Gezgin C. In the other experiments, failures are injected by human intervention, and the performance of the robot on detecting these failures and learning from them is investigated. To see the effects of background knowledge on the hypotheses, the following rules are inserted into the KB:

$$locX(A, X) \wedge 0 \leq X < 1 \Rightarrow location(A, room_1)$$
$$locX(A, X) \wedge 1 \leq X < 2 \Rightarrow location(A, room_2)$$
$$locX(A, X) \wedge 2 \leq X < 3 \Rightarrow location(A, room_3)$$

The overall manipulation performance is analyzed on 10 sets of 10 object manipulation scenarios where the objects are randomly placed. Whenever there is a $pickUp$ failure, the robot is allowed to try once more. The overall success rate of the robot in action $pickUp$ is 97.08 %. Three inherent failures have been observed: one with the green bowling pin and two with the beach ball. The reason behind these failures is the occasional localization problems during grasping. 103 observations are taken by the robot in its 12040.6 s operation time. All of the failures could be detected by the robot.

[1] http://wiki.ros.org/rviz.

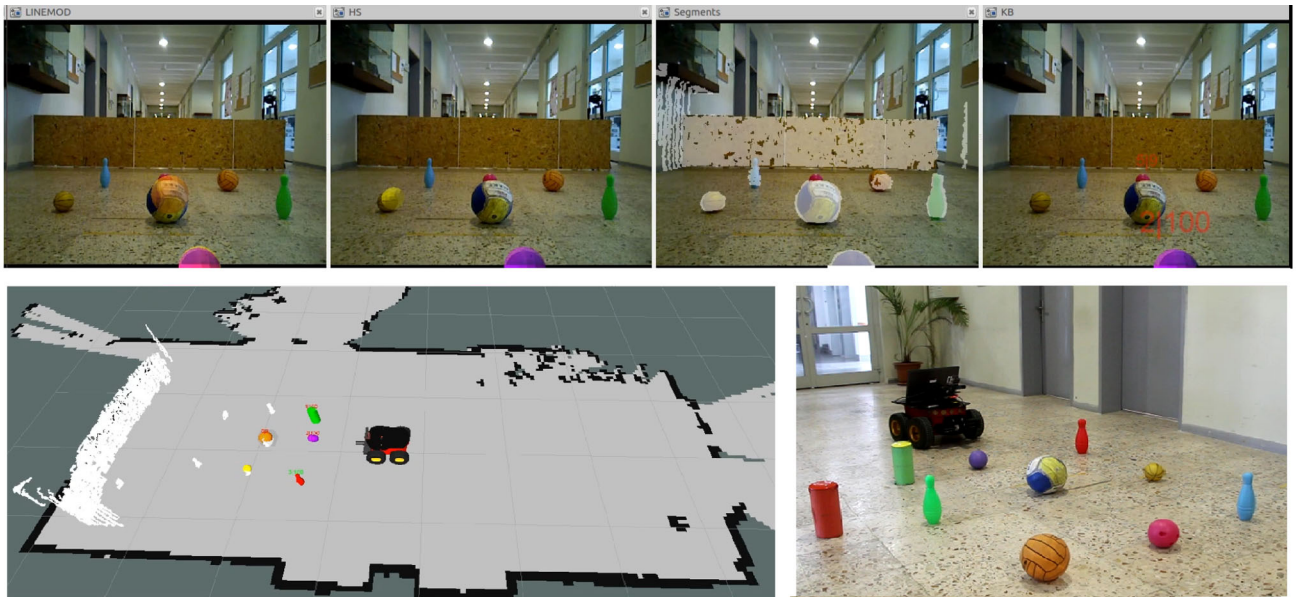


Fig. 5 Gezgin C. and the objects used in the experiments (*bottom-right*) and their representations in its world (*bottom-left*) considering 3D object recognition and segmentation results (the frames on the *top*)

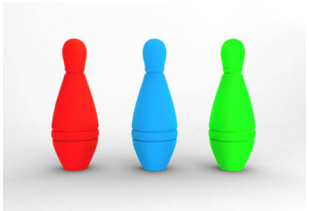


Fig. 6 es_1: failures are injected during action $pickUp$ on plastic bowling pins

When the PROGOL algorithm is applied to these observations, three hypotheses are observed to explain the failure contexts each with probabilities smaller than 0.2. The overall $putDown$ success rate is 100 %.

4.2 Learning failure contexts for a specific object category

In this experiment (es_1), Gezgin C. is enforced to fail by the intervention of a human agent during the $pickUp$ action on all plastic bowling pins (Fig. 6). The hypotheses are framed for 101 observations made in its 10769 s operation time. Three inherent failures (one with the beach ball and two with the pink ball) and 30 bowling pin failures have occurred.

The following hypotheses are framed explaining the observations:

$$\boldsymbol{h_1 : category(X, pin) \Rightarrow outcome(pickUp(X), failure)}$$
$$(P\text{: } \mathbf{1.00}, p_1 = \mathbf{30}, n_1 = \mathbf{0})$$

$$h_2 : shape(X, spherical) \wedge color(X, yellow)$$
$$\Rightarrow outcome(pickUp(X), failure)$$
$$(P\text{: } 0.07,\ p_2 = 1, n_2 = 14)$$

Table 1 es_1: Replanning analysis with and without learning

Criterion	w/o learning	w/ learning
# of search nodes	1109	179
# of extended nodes	413	77
Total planning time (s)	0.13	0.05
# of objects w/o failures	4	7
# of replans	2	0
# of failures	3	0

$$h_3 : category(X, ball) \wedge color(X, pink)$$
$$\Rightarrow outcome(pickUp(X), failure)$$
$$(P\text{: } 0.17,\ p_3 = 2, n_3 = 10)$$

In the second phase of this experiment, the guidance performance of the framed hypotheses in planning is investigated. To use its experience for its future tasks, the robot automatically updates its domain of operators based on the framed hypotheses' (with $P \geq 0.5$) premise parts as failure contexts. For guidance, the robot uses precondition update since it believes that the failures are unsafe.

We have analyzed the outcome of learning in a new planning task of collecting 10 objects with a time constraint of 10 min. External failures for bowling pins are also enforced again to simulate the failure cases. The results of the analysis are presented in Table 1 (Sariel et al. 2015). The scenario is run with plans generated by the TLPLan planner (Bacchus and Ady 2001) on both the original domain and the updated domain after learning. In these scenarios, when the robot fails in execution, it replans in its allowed time frame.

Table 2 es_1 - Learning algorithms' performance on learning pin failures

Algorithm	Accuracy	Precision	F-score
Naive Bayes	**97.09**	**0.97**	**0.99**
Bayes Network	**97.09**	**0.97**	**0.99**
SVM	**97.09**	**0.97**	**0.99**
ID3	**97.09**	**0.97**	**0.99**
PROGOL (w/o $\mathcal{B}$)	89.05	0.92	0.94
PROGOL (w/ $\mathcal{B}$)	93.06	0.93	0.96
PROGOL$-P$ (w/o $\mathcal{B}$)	**97.09**	**0.97**	**0.99**
PROGOL$-P$ (w/ $\mathcal{B}$)	**97.09**	**0.97**	**0.99**

Best values are presented in bold

As can be seen from the table, the robot using its experience never tries to manipulate the objects for which it believes it will fail. This helps in reducing the size of the total state space in planning and the planning time accordingly. The number of objects collected in the given time frame is also higher with learning.

We have also analyzed the performance of the PROGOL algorithm compared to the other supervised learning approaches (Naive Bayes Classifier, Bayes Networks, Support Vector Machines and ID3 Decision Tree Learning) with 10-fold cross validation on the observation set. Table 2 presents these results. PROGOL$-P$ results show the performance when just the hypotheses with probabilities greater than 0.5 are taken into account, and indicate that hypotheses with lower probabilities should be discarded for more correct conclusions and for eliminating noise. These results also prevail that the background knowledge ($\mathcal{B}$) on locations has no impact on the performance since it is irrelevant for the failure case. As can be seen from the table, all supervised learning methods can correctly classify the samples with 97.09 % accuracy meaning that they can determine that the failure context includes the $category - pin$ relation.

4.3 Learning failure contexts for a spatial region

In this experiment (es_2), Gezgin C. is enforced to fail by the intervention of a human agent during action $pickUp$ when the objects are located in a bounded spatial region ($room_3$, Fig. 7). The hypotheses are framed for 102 observations in its 10633.7 s operation time.

Besides the 29 external failures in the specific region ($room_3$), three inherent failures on cylindrical objects (red and green) have occurred. Without background knowledge ($\mathcal{B}$), ten different hypotheses are framed. Nine of them are with probabilities smaller than 0.5 and only one with probability 0.5 ($category(X, pin) \wedge color(X, red) \Rightarrow outcome(pickUp(X), failure)$). However, these observation contexts do not correctly explain the underlying failure

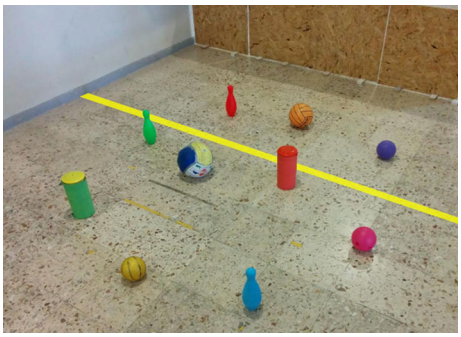


Fig. 7 es_2: failures are injected during action $pickUp$ when the objects are located in the bounded spatial region, $room_3$

situation. When $\mathcal{B}$ is incorporated, the following hypotheses are framed:

$$\boldsymbol{h_1 : location(X, room_3) \Rightarrow outcome(pickUp(X), failure)}$$
$$(P\text{: } \mathbf{1.00},\ p_1 = \mathbf{29},\ n_1 = \mathbf{0})$$

$$h_2 : category(X, cylindricalObj) \wedge color(X, red) \wedge location(X, room_2) \Rightarrow outcome(pickUp(X), failure)$$
$$(P\text{: } 0.13,\ p_2 = 1,\ n_2 = 7\ )$$

$$h_3 : category(X, cylindricalObj) \wedge color(X, green) \Rightarrow outcome(pickUp(X), failure)$$
$$(P\text{: } 0.22,\ p_3 = 2,\ n_3 = 7)$$

Among these hypotheses, the first one ($P > 0.5$) is used to guide the planner in the second scenario of collecting objects. Again, the number of objects collected by using experience is higher than that of the scenarios without learning. Furthermore, the performance is better (two more objects can be collected and the planning cost is lower.) with background knowledge, as expected (Sariel et al. 2015).

Table 3 presents the results of the learning algorithms for this experiment. As can be seen from the table, the other supervised learning algorithms cannot determine the pattern for the region in which failures occur. However, PROGOL$-P$ can incorporate background knowledge and can abstract the failure context with 97.06 % accuracy outperforming all others. This experiment set prevails the success of the ILP learning on using the background knowledge. The other supervised learning algorithms has no way of incorporating knowledge other than having a special attribute for the specific predicate.

4.4 Learning failure contexts for both a specific object category and a spatial region

In this experiment (es_3), Gezgin C. is enforced to fail by the intervention of a human agent during the $pickUp$ action on all cylindrical objects and when it interacts with the objects located in $room_3$ (Fig. 8). The hypotheses are framed for 107 observations made in its 10134 s operation time.

Table 3 es_2- Learning algorithms' performance on learning failures in a spatial region

Algorithm	Accuracy	Precision	F-score
Naive Bayes	57.28	0.66	0.73
Bayes Network	55.34	0.65	0.71
SVM	68.93	0.69	0.82
ID3	68.93	0.69	0.82
PROGOL (w/o $\mathcal{B}$)	53.40	0.63	0.70
PROGOL (w/ $\mathcal{B}$)	84.50	0.91	0.92
PROGOL$-P$ (w/o $\mathcal{B}$)	68.93	0.69	0.82
PROGOL$-P$ (w/ $\mathcal{B}$)	**97.06**	**0.97**	**0.99**

Best values are presented in bold

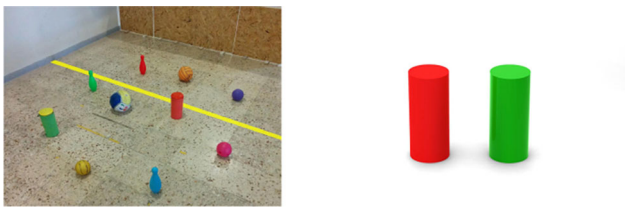


Fig. 8 es_3: failures are injected during action $pickUp$ in $room_3$ and on cylindrical objects

Besides the 32 external failures in the specific region ($room_3$) and 14 cylindrical object failures taking place outside of $room_3$ (by enforcement), seven inherent failures have occurred.

Without background knowledge ($\mathcal{B}$), four different hypotheses are framed. The following hypotheses are framed without $\mathcal{B}$:

$\boldsymbol{h_1 : deformability(X, solid)}$
$\boldsymbol{\Rightarrow outcome(pickUp(X), failure)}$
(P: **0.53**, $p_1 = \mathbf{42}$, $n_1 = \mathbf{38}$)

$h_2 : shape(X, spherical) \wedge color(X, yellow)$
$\Rightarrow outcome(pickUp(X), failure)$
(P: 0.30, $p_2 = 3$, $n_2 = 7$)

$\boldsymbol{h_3 : category(X, bigBall) \wedge color(X, FB)}$
$\boldsymbol{\Rightarrow outcome(pickUp(X), failure)}$
(P: **0.50**, $p_3 = \mathbf{5}$, $n_3 = \mathbf{5}$)

$h_4 : category(X, bigBall) \wedge color(X, orange)$
$\Rightarrow outcome(pickUp(X), failure)$
(P: 0.43, $p_4 = 3$, $n_4 = 4$)

These observation contexts do not correctly explain the underlying failure situations. When $\mathcal{B}$ is incorporated, the following hypotheses are framed:

$\boldsymbol{h_1 : location(X, room_3) \Rightarrow outcome(pickUp(X), failure)}$
(P: **1.00**, $p_1 = \mathbf{32}$, $n_1 = \mathbf{0}$)

$\boldsymbol{h_2 : category(X, cylindricalObj)}$
$\boldsymbol{\Rightarrow outcome(pickUp(X), failure)}$
(P: **1.00**, $p_2 = \mathbf{14}$, $n_2 = \mathbf{0}$)

Table 4 es_3-Learning algorithms' performance on learning contexts related to both a specific object category and a spatial region

Algorithm	Accuracy	Precision	F-score
Naive Bayes	62.62	0.68	0.77
Bayes Network	66.36	0.68	0.80
SVM	69.16	0.69	0.82
ID3	65.42	0.68	0.79
PROGOL (w/o $\mathcal{B}$)	50.50	0.64	0.73
PROGOL (w/ $\mathcal{B}$)	85.00	0.92	0.92
PROGOL$-P$ (w/o $\mathcal{B}$)	59.81	0.65	0.75
PROGOL$-P$ (w/ $\mathcal{B}$)	**91.59**	**0.96**	**0.96**

Best values are presented in bold

$h_3 : color(X, green) \Rightarrow outcome(pickUp(X), failure)$
(P: 0.11, $p_3 = 1$, $n_3 = 8$)

$h_4 : category(X, bigBall) \wedge location(X, room_1)$
$\Rightarrow outcome(pickUp(X), failure)$
(P: 0.43, $p_4 = 3$, $n_4 = 4$)

$h_5 : color(X, FB) \wedge location(X, room_1)$
$\Rightarrow outcome(pickUp(X), failure)$
(P: 0.33, $p_5 = 2$, $n_5 = 4$)

$h_6 : category(X, bigBall) \wedge color(X, orange)$
$\Rightarrow outcome(pickUp(X), failure)$
(P: 0.20, $p_6 = 1$, $n_6 = 4$)

The first two hypotheses ($P > 0.5$) are to be used to guide the planner (Sariel et al. 2015). When the results of the learning algorithms are compared (Table 4), it can be seen again that the other supervised learning algorithms fail in recognizing the patterns in observation contexts with outcome $failure$. However, PROGOL$-P$ can incorporate background knowledge and can abstract the failure context with 91.59 % accuracy outperforming all others.

4.5 Learning failure contexts for the tabletop object manipulation domain

The main superiority of the ILP-based learning algorithm to conventional machine learning algorithms is its knowledge-based representation. It can represent hypotheses in FOL and incorporate background knowledge. To illustrate why it is better in our system, we have set up another experiment (es_4), and analyzed the performance of the learning algorithm on how well it generalizes to frame more abstract conclusions. For this purpose, we have used our robot arm and set up an experiment in the blocks world domain. In this domain, given an initial configuration of the blocks, the robot generates a plan to build the given target structure. We focused on the 3-block stacking scenario in this domain, and let the robot work with two sets of blocks (both 4cm-edge $narrow$ blocks, 5.8cm-edge $wide$ blocks graspable by the gripper). The primitive behaviors of $move$ the end effector to a global position or

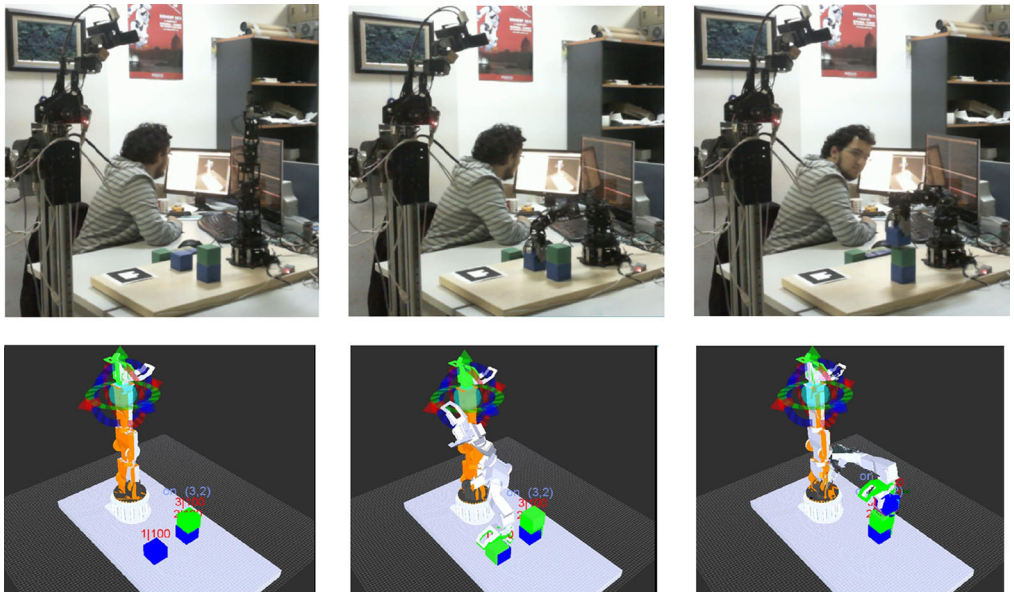


Fig. 9 3-block stacking scenario. The *top* row presents the phases of action *stack* execution in the real world (from left to right). The *bottom* row represents the robot's world model in Rviz. The recognized objects along with their ids and confidence values, and the relations among them (e.g.,$on(obj_3, obj_2)$) can be tracked online

to the position of an object, $pickUp$ and $putDown$ behaviors are designed on the robot system. By executing these primitive behaviors, the robot can *stack* objects on top of each other. To determine the initial state, the objects in the scene are recognized and their relations are determined by *Scene Interpretation*. We have used an ARTag label placed on the table as a reference point to transform and calculate the global positions of the objects with respect to the position of the robot arm. We have used MoveIt[2] framework for kinematic computations and safe trajectory planning to avoid from obstacles and the other objects. The robot continually executes $move - pickUp - transport - stack$ actions to build a 3-block stack structure. The blocks can take on two different colors: $green$ and $blue$. An illustration from a sample execution scenario with wide blocks is given in Fig. 9. Videos of the sample scenarios are also available online[3].

The attributes of objects and their relations are encoded in each observation. A sample observation is given as follows:

$$
\begin{aligned}
&holding(obj_2) \wedge on\_table(obj_1) \wedge clear(obj_1)\\
&\wedge category(obj_1, block) \wedge material(obj_1, wooden)\\
&\wedge shape(obj_1, cubic) \wedge deformability(obj_1, solid)\\
&\wedge color(obj_1, blue) \wedge width(obj_1, narrow) \wedge height(obj_1, short)\\
&\wedge locX(obj_1, -0.083) \wedge locY(obj_1, -0.202) \wedge locZ(obj_1, -0.060)\\
&\wedge category(obj_2, block) \wedge material(obj_2, wooden)\\
&\wedge shape(obj_2, cubic) \wedge deformability(obj_2, solid)\\
&\wedge color(obj_2, blue) \wedge width(obj_2, narrow) \wedge height(obj_2, short)\\
&\wedge locX(obj_2, -0.085) \wedge locY(obj_2, -0.204) \wedge locZ(obj_2, -0.021)\\
&outcome(stack(obj_2, obj_1), failure)
\end{aligned}
$$

We first analyzed the $pickUp$ success of the robot arm and observed that the overall success rate is 84.85 %. Note that no failure is injected into any blocks world domain experiment. Just by considering these observations, the following hypotheses are framed for explaining failure contexts in action $pickUp$:

$$
\begin{aligned}
&h_1 : color(X, green) \wedge width(X, narrow)\\
&\Rightarrow outcome(pickUp(X), failure)\\
&(P\text{: } 0.18,\ p_1 = 3,\ n_1 = 14)\\
&h_2 : color(X, blue) \wedge width(X, narrow)\\
&\Rightarrow outcome(pickUp(X), failure)\\
&(P\text{: } 0.13,\ p_2 = 2,\ n_2 = 13)\\
&h_3 : color(X, green) \wedge width(X, wide)\\
&\Rightarrow outcome(pickUp(X), failure)\\
&(P\text{: } 0.07,\ p_3 = 1,\ n_3 = 14)\\
&h_4 : color(X, blue) \wedge width(X, wide)\\
&\Rightarrow outcome(pickUp(X), failure)\\
&(P\text{: } 0.22,\ p_4 = 4,\ n_4 = 14)
\end{aligned}
$$

Then, the $stack$ performance is measured in which the robot is always successful in picking up an object (otherwise, stacking is not possible). The following hypotheses are framed for explaining failure contexts in action $stack$ without any background knowledge $\mathcal{B}$:

$$
\begin{aligned}
&h_1 : on\_table(B) \wedge color(A, green) \wedge width(A, narrow)\\
&\Rightarrow outcome(stack(A, B), failure)\\
&(P\text{: } 0.29,\ p_1 = 2,\ n_1 = 5)\\
&h_2 : color(A, green) \wedge width(A, narrow)\\
&\Rightarrow outcome(stack(A, B), failure)\\
&(P\text{: } 0.25,\ p_2 = 2,\ n_2 = 6)\\
&h_3 : color(A, blue) \wedge width(A, narrow)\\
&\Rightarrow outcome(stack(A, B), failure)\\
&(P\text{: } 0.23,\ p_3 = 3,\ n_3 = 10)
\end{aligned}
$$

[2] http://moveit.ros.org.

[3] http://air.cs.itu.edu.tr/tubitak-111e286.

Here, the first hypothesis covers the cases where the robot fails in executing a stack action for placing green and narrow blocks in the second level. The second hypothesis, on the other hand, correspond to cases for stacking both in the second or the third levels. But without background knowledge, it is not possible to infer any relation with the stack position. The third hypothesis is a general hypothesis explaining cases where the block to be stacked is in color $blue$ and its width is $narrow$.

To see how the use of background knowledge affects the generalization performance of the PROGOL-P algorithm, we inserted the following $\mathcal{B}$ rules on tower levels into the KB.

$$on\_table(X) \wedge clear(X) \Rightarrow tower(X, 1)$$
$$on(X, Y) \wedge tower(Y, N) \wedge clear(X) \Rightarrow tower(X, N + 1)$$

Considering the observations and the background knowledge, the robot can frame the following hypotheses:

$$h_2 : tower(B, 1) \wedge color(A, blue) \wedge width(A, narrow) \Rightarrow outcome(stack(A, B), failure)$$
$$(P: 0.29,\ p_1 = 2, n_1 = 5)$$

$$h_2 : tower(B, 1) \wedge color(A, blue) \wedge width(A, narrow) \Rightarrow outcome(stack(A, B), failure)$$
$$(P: 0.14,\ p_4 = 1, n_4 = 6)$$

$$h_3 : tower(B, 2) \wedge color(A, green) \wedge width(A, narrow) \Rightarrow outcome(stack(A, B), failure)$$
$$(P: 0.40,\ p_2 = 2, n_2 = 3)$$

$$h_4 : tower(B, 2) \wedge color(A, blue) \wedge width(A, narrow) \Rightarrow outcome(stack(A, B), failure)$$
$$(P: 0.29,\ p_3 = 2, n_3 = 5)$$

As can be seen from this result, the robot believes that it is more likely (failure probability, P = 0.33) to fail in the case of building a three-level tower structure with narrow blocks while building a two-level tower structure is more probable (failure probability, P = 0.21). Note that the color attribute is taken into account as $color(A, blue) \vee color(A, green)$ since these are the only colors that are observed. When more observations are provided, the learning algorithm can find this attribute irrelevant for the context representation. To see this effect, we produced synthetic data with sample observations on stacking objects with another color in the third level. We have seen that after four observations, the hypothesis is generalized as: $tower(B, 2) \wedge width(A, narrow) \Rightarrow outcome(stack(A, B), failure)$. However, if the failure is due to object recognition, the failure becomes persistent for objects with a specific color, and this attribute takes an important role in that case.

Obviously, an attribute-based learner cannot determine such relations easily. It needs to encode all pairwise relations instead. For this specific example, the real cause of the failure may be related to a vision problem, the instability of the robot's arm after a certain height or displacement of the blocks in the horizontal line leading to imbalance. If there is no quantitative or qualitative measurement way for these issues, it is difficult to isolate the failure. However, even when the actual underlying reason cannot be identified, the robot believes that it will fail in executing action *stack* to put on a block on top of a tower with two or more levels, and then will plan accordingly in its future tasks.

5 Discussion

Our experiments ($es_1 - es_3$) in the ground manipulation domain verify the success of our experience-based learning approach in extracting patterns in observation contexts for failure cases. The first experiment es_1 involves an easy scenario for all learning algorithms to correctly extract the failure pattern as a specific object category. However, es_2 and es_3 are challenging experiment sets in which standard machine learning algorithms fail in finding failure patterns. Our learning algorithm, on the other hand, makes use of background rules $\mathcal{B}$ which helps in the abstracted representation of certain spatial regions in the environment, thus, outperforming the other methods. This property is more useful with more complex scenarios when a conclusion needs to be made by forward chaining.

The supervised learning algorithms need to represent each predicate that can be concluded as a new attribute, which is not scalable for even modest-size knowledge bases. The strength of the knowledge-based learning in this regard is made clear in the tabletop experiment with the robot arm (es_4). Manipulation capabilities of the robot arm, which can build more complex structures, are broader than that of the ground robot. Therefore, this robot can execute a block stacking scenario. By the use of relational predicates in the background theory $\mathcal{B}$ on tower description for the block stacking scenario, the observation context for the failure case can be generalized and mapped to the height of the tower rather than representing only the attributes of the blocks. The use of background knowledge allows the robot to learn relational predicates and extract more general hypotheses. This is also useful for inter-domain experience transfer.

The main superiority of our learning approach lies in that even when the underlying reasons of failures are not given, relations can be learned based on a sufficient number of observations. This is validated by all the experiments.

Due to limitations in sensing, there may be missing or incorrect predicates in observation contexts. Although most of them are eliminated by *Scene Interpreter*, observations may include noise. This drawback is alleviated by probabilistic representation of hypotheses in our learning algorithm. As a future work, more emphasis can be placed on the appraisal of observations and incremental construction of hypotheses.

It is important for robots to build their experience online, and then to use this experience for guidance in task execution. The experiments show that our system automatically generates relevant domain constraints based on the gained experience online, and these constraints are applied in new planning tasks. As we would expect and see from the results, this results in the generation of safer plans. As a positive side effect, the size of the search space is also reduced during planning.

The existing implementation of the learning algorithm uses background knowledge $\mathcal{B}$ as a conjunction of hand-made rules provided externally. In the future, it is possible that the robot connects to existing knowledge bases and ontologies as in (Saxena et al. 2014; Tenorth and Beetz 2013; Waibel et al. 2011) to apply reasoning patterns. This improves the strength of the algorithm in generalization of the derived rules.

With the existing *Scene Interpreter*, categories of objects are determined based on template matching. Color and shape can be extracted automatically. However, the other attributes are predetermined and associated with the templates. Automated attribute extraction methods can be developed by more advanced vision algorithms that may also depend on the improvements on the technologies of vision sensors.

In the derivation of hypotheses, the full set of observed or known predicates are used as observations. Only relevant features can be considered in observation contexts for reducing the complexity. An interesting future direction would be guiding the control of robots in getting these relevant features only or guiding them in making the right observations to derive correct hypotheses.

6 Conclusion

We have presented our ILP-based method for real-world experience-based learning in cognitive robots. This method ensures that robots learn action execution failure contexts as hypotheses, and use them for ensuring safety in their future tasks. In hypotheses, the observable attributes of and the relations among the objects and the relevant facts of the world are represented in FOL which is useful for reasoning and planning. Partially specified world states can also be easily represented by only the observable predicates. Conclusion parts of hypotheses are related to failure and success cases. We have applied a probabilistic hypothesis framing scheme to deal with both noisy sensory data and inconsistencies. Our results on real-world scenarios indicate that our learning method can frame correct hypotheses on failure contexts. Although ILP-based learning is not the perfect standalone supervised learning approach for classification tasks, it has been observed that its use of background knowledge makes it a suitable candidate for creating robot intelligence in learning tasks. It can generalize well using background theory effectively and applying reasoning to make abstract conclusions. The other supervised learning methods can not easily incorporate background knowledge. Since the ILP framework can use partial specifications of the world states, it can deal with the missing data problem. In the current implementation, all the observable predicates are taken into account during learning. Our future work includes automatic selection of relevant predicates only.

Acknowledgments This research is funded by a grant from the Scientific and Technological Research Council of Turkey, Grant No. 111E-286. We thank Petek Yildiz for her contribution in the planning guidance implementations, Burak Topal and Abdullah Cihan Ak for their contributions in the controller design of the robot arm and help in the experiments, Dogan Altan for ground robot experiments, Mustafa Ersen, Melis Kapotoglu, Melodi Deniz Ozturk, Cagatay Koc and Arda Inceoglu for their contributions in different components of the framework.

References

Aler, R., Borrajo, D., & Isasi, P. (2002). Using genetic programming to learn and improve control knowledge. *Artificial Intelligence*, *141*, 141–1.

Aoude, G. S., Luders, B. D., Levine, D. S., & How, J. P. (2010) Threat-aware path planning in uncertain urban environments. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*.

Bacchus, F., & Ady, M. (2001). Planning with resources and concurrency a forward chaining approach. In *Proceedings of the 17th international joint conference on artificial intelligence* (Vol. 1, pp. 417–424).

Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, *116*(1), 123–191.

Borrajo, D., & Veloso, M. (1994). Incremental learning of control knowledge for improvement of planning efficiency. In *AAAI-94 fall symposium on planning and learning* (pp. 5–9).

Borrajo, D., & Veloso, M. (1996). Lazy incremental learning of control knowledge for efficiently obtaining quality plans. AI Review Journal. Special Issue on Lazy Learning **11**, 371–405.

Botea, A., Enzenberger, M., Müller, M., & Schaeffer, J. (2005). Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research*, *24*, 581–621.

Chow, Y. L., Pavone, M., Sadler, B. M., & Carpin, S. (2014). Trading safety versus performance: Rapid deployment of robotic swarms with robust performance constraints. *Journal of Dynamic Systems, Measurement and Control*, *137*(3), 031005.

Cohen, W. W. (1990). Learning approximate control rules of high utility. In *Proceedings of the seventh international conference on machine learning*, Morgan Kaufmann, Burlington (pp. 268–276).

Dearden, R., & Burbridge, C. (2014). Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, *62*(3), 355–365.

Duran, G. (2006) Integrating macro-operators and control-rules learning. In *Proceedings of the international conference on automated planning and scheduling (ICAPS)*, Cumbria, UK.

Ersen, M., Ozturk, M. D., Biberci, M., Sariel, S., & Yalcin, H. (2014) Scene interpretation for lifelong robot learning. In *Proceedings*

of the 9th international workshop on cognitive robotics (CogRob 2014) held in conjunction with ECAI-2014, Prague, Czech Republic.

Ersen, M., Sariel-Talay, S., & Yalcin, H. (2013). Extracting spatial relations among objects for failure detection. In *Procedings of the KI 2013 workshop on visual and spatial cognition* (pp. 13–20).

Estlin, T. A., & Mooney, R. J. (1997) Learning to improve both efficiency and quality of planning. In *Proceedings of the fifteenth international joint conference on artificial intelligence*, Morgan Kaufmann, Burlington (pp. 1227–1232).

Fernandez, S., Aler, R., & Borrajo, D. (2004). Using previous experience for learning planning control knowledge. In *Proceedings of the seventeen international Florida artificial intelligence symposium (FLAIRS04)*, AAAI Press.

Feyzabadi, S., Carpin, S. (2014). Risk-aware path planning using hirerachical constrained markov decision processes. In *Proceedings of the IEEE international conference on automation science and engineering (CASE)*.

Gspandl, S., Pill, I., Reip, M., Steinbauer, G., & Ferrein, A. (2011). Belief management for high-level robot programs. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)*.

Haigh, K. Z., & Veloso, M. M. (1999). Learning situation-dependent costs: Improving planning from probabilistic robot execution. *Robotics and Autonomous Systems*, *29*, 145–174.

Hermans, T. R. (2014). Representing and learning affordance-based behaviors. Ph.D. thesis, Georgia Institute of Technology.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P. F., Navab, N., Fua, P., et al. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *34*(5), 876–888.

Horn, A. (1951). On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, *16*(1), 14–21.

Kabanza, F., Barbeau, M., & St-Denis, R. (1997). Planning control rules for reactive agents. *Artificial Intelligence*, *95*(1), 67–113.

Kapotoglu, M., Koc, C., Sariel, S., & Ince, G. (2014) Action monitoring in cognitive robots. In *Proceedings of the IEEE 22nd signal processing and communications applications conference (SIU)* (pp. 2154–2157).

Karapinar, S., Altan, D., & Sariel-Talay, S. (2012) A robust planning framework for cognitive robots. In *Proceedings of the AAAI-12 workshop on cognitive robotics (CogRob)* (pp. 102–108).

Karapinar, S., Ersen, M., Kapotoglu, M., Yildiz, P., Sariel-Talay, S., & Yalcin, H. (2013). Bilissel robotlarda deneyimsel ogrenme. In *Proceedings of the 21st IEEE signal processing and communications applications conference (SIU)*, Girne, KKTC.

Karapinar, S., & Sariel, S. (2015). Cognitive robots learning failure contexts through experimentation. In *Proceedings of the 14th international conference on autonomous agents & multiagent systems (AAMAS)*.

Karapinar, S., & Sariel-Talay, S. (2012). Failure handling in a planning framework. In *Proceedings of the AAAI-12 student workshop*, AAAI Press, Toronto, ON, Canada (pp. 2431–2432).

Karapinar, S., Sariel-Talay, S., Yildiz, P., & Ersen, M. (2013). Learning guided planning for robust task execution in cognitive robotics. In *Proceedings of the AAAI-13 workshop on intelligent robotic systems*, Bellevue, USA (pp. 26–31).

Katukam, S., & Kambhampati, S. (1994). Learning explanation-based search control rules for partial order planning. In *Proceedings of the 12th national conference on artificial intelligence (AAAI)* (pp. 582–587).

Khardon, R. (1997). Learning action strategies for planning domains. *Artificial Intelligence*, *113*, 125–148.

Kolb, D. A. (1984). *Experiential learning: Experience as the source of learning and development*. Englewood Cliffs: Prentice-Hall P T R.

Kvarnström, J., Heintz, F., & Doherty, P. (2008). A temporal logic-based planning and execution monitoring system. In *Proceedings of the 18th international conference on automated planning and scheduling (ICAPS)* (pp. 198–205).

Lang, T., & Toussaint, M. (2010). Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, *39*, 1–49.

Leckie, C., & Zukerman, I. (1998). Inductive learning of search control rules for planning. *Artificial Intelligence*, *101*(1–2), 63–98.

López, B., & Plaza, E. (1991) Case-based learning of strategic knowledge. In *Proceedings of the European working session on learning porto (EWSL)* (pp. 398–411)

Magnenat, S., Chappelier, J. C., Mondada, F. (2012). Integration of online learning into htn planning for robotic tasks. In *Proceedings of the AAAI spring symposium: Designing intelligent robots, AAAI technical report, AAAI* (Vol. SS-12-02).

Morisset, B., & Ghallab, M. (2008). Learning how to combine sensory-motor functions into a robust behavior. *Artificial Intelligence*, *172*(4), 392–412.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*(3–4), 245–286. Special issue on Inductive Logic Programming.

Ozturk, M. D., Ersen, M., Kapotoglu, M., Koc, C., Sariel-Talay, S., & Yalcin, H. (2014). Scene interpretation for self-aware cognitive robots. In *Proceedings of the AAAI spring symposia qualitative representations for robots*, California, USA.

Pasula, H., Zettlemoyer, L. S., & Kaelbling, L. P. (2007). Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, *29*, 309–352.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009) ROS: an open-source robot operating system. In *Proceedings of the ICRA-09 workshop on open source software*.

Rintanen, J. (2000). Incorporation of temporal logic control into plan operators. In *Proceedings of the European conference on artificial intelligence (ECAI)*, IOS Press (pp. 526–530).

Sariel, S., Yildiz, P., Karapinar, S., Altan, D., & Kapotoglu, M. (2015). Robust task execution through experience-based guidance for cognitive robots. In *Proceedings of the 17th international conference on advanced robotics (ICAR)*.

Saxena, A., Jain, A., Sener, O., Jami, A., Misra, D. K., & Koppula, H. S. (2014). RoboBrain: Large-scale knowledge engine for robots. Computing Research Repository (CoRR) abs/1412.0691

Stansbury, R. S., & Agah, A. (2012). A robot decision making framework using constraint programming. *Artificial Intelligence Review*, *38*, 67–83.

Tenorth, M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics Research*, *32*(5), 566–590.

Trevor, A. J. B., Gedikli, S., Rusu, R. B., & Christensen, H. I. (2013) Efficient organized point cloud segmentation with connected components. In *Proceedings of the 3rd workshop on semantic perception, mapping and exploration (SPME)*.

Usug, U. C., Altan, D., & Sariel-Talay, S. (2012). Robots that create alternative plans against failures. In *Proceedings of the 10th IFAC symposium on robot control*.

Usug, U. C., & Sariel-Talay, S. (2011). Dynamic temporal planning for multirobot systems. In *Proceedings of the AAAI-11 workshop on automated action planning for autonomous mobile robots (PAMR)*.

Waibel, M., Beetz, M., Civera, J., DÁndrea, R., Elfring, J., Galvez-Lopez, D., et al. (2011). Roboearth—A world wide web for robots. *IEEE Robotics Automation Magazine*, *18*(2), 69–82.

Wang, X., Simon, H. A., Lehman, J. F., & Fisher, D. H. (1996). Learning planning operators by observation and practice. In *Proceedings of the second international conference on AI planning systems, AIPS-94* (pp. 335–340).

Yildiz, P., Karapinar, S., & Sariel-Talay, S. (2013). Learning guided symbolic planning for cognitive robots. In *Proceedings of the IEEE international conference on robotics and automation (ICRA), autonomous learning workshop*, Karlsruhe, Germany.

Sertac Karapinar received the B.S. and M.Sc. degrees in computer engineering from Istanbul Technical University (ITU), Istanbul, Turkey, in 2010 and 2013, respectively. He is now a PhD candidate at ITU. His research interests include cognitive robots, robot learning and bio-inspired intelligence.

Sanem Sariel received the B.S., M.Sc., and Ph.D. degrees in computer engineering from Istanbul Technical University (ITU), Istanbul, Turkey, in 1999, 2002, and 2007, respectively. During her Ph.D. studies, she worked as a Researcher at Georgia Institute of Technology, Atlanta from 2004 to 2006. She is currently an Assistant Professor at ITU. Her research interests include cognitive robots, multirobot systems and artificial intelligence in games.