HAL open science

# Leveraging Multiple Environments for Learning and Decision Making: a Dismantling Use Case

Alejandro Suárez-Hernández, Thierry Gaugry, Javier Segovia-Aguas, Antonin Bernardin, Carme Torras, Maud Marchal, Guillem Alenyà

**To cite this version:**

Alejandro Suárez-Hernández, Thierry Gaugry, Javier Segovia-Aguas, Antonin Bernardin, Carme Torras, et al.. Leveraging Multiple Environments for Learning and Decision Making: a Dismantling Use Case. IROS 2020 - IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2020, Las Vegas / Virtual, United States. pp.6902-6908. hal-03132986

**HAL Id: hal-03132986**

**https://inria.hal.science/hal-03132986v1**

Submitted on 5 Feb 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Leveraging Multiple Environments for Learning and Decision Making: a Dismantling Use Case

Alejandro Suárez-Hernández[1] and Thierry Gaugry[2] and Javier Segovia-Aguas[1]
and Antonin Bernardin[2] and Carme Torras[1] and Maud Marchal[2] and Guillem Alenyà[1]

***Abstract*— Learning is usually performed by observing real robot executions. Physics-based simulators are a good alternative for providing highly valuable information while avoiding costly and potentially destructive robot executions. We present a novel approach for learning the probabilities of symbolic robot action outcomes. This is done leveraging different environments, such as physics-based simulators, in execution time. To this end, we propose MENID (*Multiple Environment Noise Indeterministic Deictic*) rules, a novel representation able to cope with the inherent uncertainties present in robotic tasks. MENID rules explicitly represent each possible outcomes of an action, keep memory of the source of the experience, and maintain the probability of success of each outcome. We also introduce an algorithm to distribute actions among environments, based on previous experiences and expected gain. Before using physics-based simulations, we propose a methodology for evaluating different simulation settings and determining the least time-consuming model that could be used while still producing coherent results. We demonstrate the validity of the approach in a dismantling use case, using a simulation with reduced quality as simulated system, and a simulation with full resolution where we add noise to the trajectories and some physical parameters as a representation of the real system.**

## I. Introduction

Probabilistic propositional planning [1] consists in computing behavioral policies reasoning over a set of actions whose outcomes are uncertain. This policy is meant to maximize a reward score. Since reward maximization subsumes goal satisfaction, probabilistic planning can be considered an extension over classical AI planning. Therefore, probabilistic planning can be regarded as a more powerful tool for handling the kind of task that typically arise in robotics. In this paper, we focus on the problem of learning the categorical distribution associated to the outcomes of a stochastic symbolic action, with the support of simulations.

Reinforcement Learning (RL) usually requires exhaustive interaction with the environment and results in less explainable reasoning. The work of Eppe *et al.* [2] tackles these issues through the combination of a symbolic planner to set the subgoal roadmap, and a RL module to decide the specifics on how each subgoal is achieved. They do not tackle, however, the challenge of learning the distributions of symbolic stochastic actions.

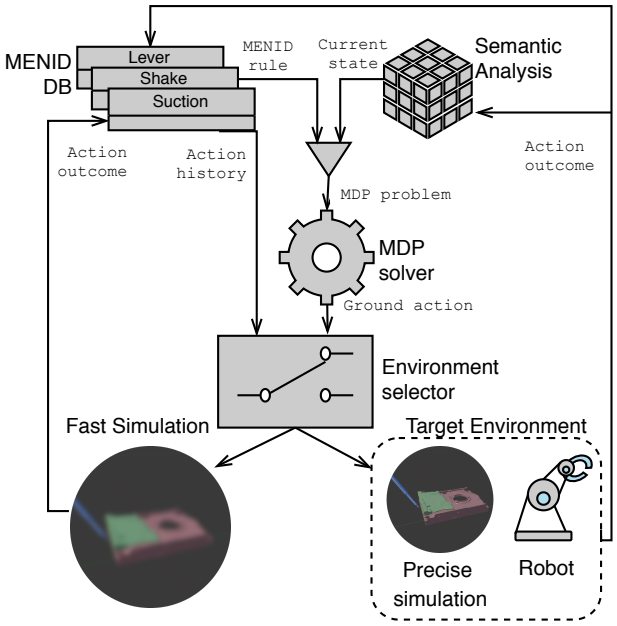



Fig. 1. System's architecture. The action database contains MENID rules updated with experiences from both the simulated and the target environments. In turn, the execution history of the action is used for optimal decision making. The environment selection criteria for a given action depends on: (1) the simulator's accuracy; and (2) the confidence in the action's empirical outcome distribution.

In pure AI planning, it is often assumed that accurate models of the actions are available. This kind of knowledge can hardly be taken for granted in many real world settings. While several approaches exist for automatically acquiring deterministic models [3]–[6], probabilistic ones require special care. Previous approaches have explored the idea of information gathering for contingent plan execution [7]–[9]. These approaches share a similar methodology: they interleave learning, reasoning and execution. However, if a robot applied this general strategy, it would be prone to put itself and its surroundings in harm's way. Lipovetzky *et al.* [10] use simulators, like us, but their premise is based on blind search without considering the actions' model.

We propose a novel knowledge gathering algorithm that distributes executions among different environments. One of these environments is the *target* environment (e.g. real world or highly accurate simulator). Executing actions in this

*The research leading to these results has received funding from the EU H2020 research and innovation programme under grant agreement no. 731761, IMAGINE; the HuMoUR project TIN2017-90086-R (AEI/FEDER, UE); and AEI through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656).

[1]Authors are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC Llorens i Artigas 4-6, 08028, Barcelona, Spain {asuarez,jsegovia,torras,galenya}@iri.upc.edu

[2]Authors are with Univ. Rennes, INSA, IRISA, Inria, France {Maud.Marchal, Thierry.Gaugry, Antonin.Bernardin}@inria.fr

environment is costly and risky. Therefore, we would like to minimize the amount of *exploratory* actions performed within it. On the other hand, we have one or more *test* environments that act as a proxy of the target one (e.g. fast simulator). Execution in a test environment is safe and, usually much less costly.

In this work, we rely on SOFA, a physics-based simulator [11] to carry out accurate predictions of the actions' outcome. With SOFA, we are able to simulate several actions varying the different physical parameters. With high-resolution models, these simulations are very accurate, although also very time-consuming. A trade-off between accuracy and computation time performance must be determined.

Our methodology is suitable to be of use in a wide range of robotic tasks. In this paper, we demonstrate it in a two-environments setting (see Fig. 1). We focus on the use case of dismantling electro-mechanical devices. This kind of problems is subject to non-determinism in the action execution, as manipulation skills (such as levering) depend on factors that escape our control (e.g. friction, jerky motions) and are somewhat unpredictable.

Our contributions are summarized as follows:

- A novel learning algorithm that decides how to gather data from two sources to learn the probabilistic outcome of high-level actions.
- MENID (*Multiple Environment Noise Indeterministic Deictic*) rules, an extension of NID rules [8] to cope with the multi-environment setting.
- A complete workflow illustrating our approach by using physics-based simulations for generating data for the learning process.

## II. Learning Methodology

We want to enable our robot to learn and perform optimal decision making in execution time. To this end, we propose an algorithm that borrows notions from probability estimation and information theory.

In this section, we present first a new action model meant to cope with several environments. Then, we outline the steps of our algorithm. Afterwards, we elaborate further on its key components.

### A. MENID rules

In the past, NID (*Noise Indeterministic Deictic*) rules have been proposed as a mechanism to model probabilistic actions [8], [9]. These rules enjoy the following advantages:

- *Deictic references*: variables that identify objects related to the parameters and are ground in the precondition.
- *A noise effect* that covers unknown outcomes or outcomes that cannot be modeled explicitly.
- *Derived predicates* that depend on the truth value of other predicates in the state and that allow expressing complex preconditions.

A rule has associated a list of outcomes, each with its probability of happening. However, this probability is bound to a single environment. To handle multiple environments, we define a new kind of rule called MENID (*Multiple Environment NID*) as follows

$$a_r(\chi) : \phi(\chi) \rightarrow \begin{cases} {}^1p_{r,1} \dots {}^ep_{r,1} & : \Omega_{r,1}(\chi) \\ \dots \\ {}^1p_{r,n_r} \dots {}^ep_{r,n_r} & : \Omega_{r,n_r}(\chi) \\ {}^1p_{r,0} \dots {}^ep_{r,0} & : \Omega_{r,0}(\chi) \end{cases}, \tag{1}$$

where

- $a_r$ is the action associated to this rule.
- $\chi$ is the set of variables for the rule, composed of two disjoint subsets: the action's parameters $\chi_a$, and the rule's deictic references $\chi_r$.
- $\phi_r(\chi)$ is the preconditon: a set of predicates that must be present in the current state for this rule to trigger. When $a_r$ is executed, only the rule whose preconditions are satisfied is applied.
- Each $\Omega_{r,i}(\chi)$ is a different possible outcome (i.e. a set of predicates that are added or deleted from the state).
- $\Omega_{r,0}(\chi)$ is the noise effect.
- Each ${}^jp_{r,i}$ represents outcome $\Omega_{r,i}(\chi)$'s probability in environment $j$ (up to $e$ environments).

An action can consist of several rules, but a rule is linked to just one action. This allows to conveniently define actions that, depending on the context, affect the world differently.

The distribution over outcomes of a rule is categorical. Therefore, outcome frequency when repeatedly triggering a rule results follows a multinomial distribution. If ${}^jx_{r,i}$ is the absolute frequency of outcome $i$ in environment $j$, and the number of times that $r$ has been triggered in $j$ is $\sum_i^{n_r} {}^jx_{r,i}$, the empirical categorical distribution is

$${}^jp_{r,i} \approx {}^jq_{r,i} = \frac{{}^jx_{r,i}}{\sum_{i=0}^{n_r} {}^jx_{r,i}}. \tag{2}$$

In the next sections, we focus on the particular case of two environments ($e = 2$), being $j = 1$ the target environment, and $j = 2$ the test one. We describe our methodology to combine both sources of information to approximate the target distribution.

### B. Algorithm Overview

Algorithms to learn NID rules from experiences already exist, particularly for the case of guided demonstrations [9] and exogenous effects [12]. We focus here on learning the different outcome probabilities ${}^jp_{r,i}$. Thus, we assume that the system is initialized with all the MENID rules representing the actions available to the robot, along with their preconditions. For each rule, a list of outcomes (postconditions) $\Omega_{r,i}(\chi)$. is also present.

The main procedure of our methodology is sketched in Algorithm 1. It has been tailored to the particular case of one target environment and one test environment. However, it can be easily extended to accommodate more than two sources. Next, we describe the details of the algorithm.

**Initialization:** Set $E$ represents the gathered experiences, and thus is initialized to the empty set (line 1).

**Algorithm 1** Learning-execution loop

**Input:** reward function $R$, action set $A = \{a_1, \ldots, a_n\}$, time allotment for action testing $T$

1: $E \leftarrow \emptyset$
2: **loop**
3: $s \leftarrow$ *get_current_state*()
4: $P_{target} \leftarrow$ target transition derived from MENID rules
5: $a \leftarrow$ optimal action in $s$ according to $P_{target}$
6: **if** $a$ not marked and $\delta > \delta_{thres}$ **then**
7: mark $a$
8: $t \leftarrow T$
9: **while** $t > 0$ **do**
10: $s' \leftarrow$ *test_action*$(s,a)$
11: *elapsed* $\leftarrow$ time spent testing $a$
12: $E \leftarrow E \cup \{(test, s, a, s')\}$
13: $t \leftarrow t - elapsed$
14: **end while**
15: **else**
16: remove mark from $a$
17: $s' \leftarrow$ *exec_action*$(a)$
18: $E \leftarrow E \cup \{(target, s, a, s')\}$
19: **end if**
20: Update MENID rules according to $E$
21: **end loop**

**Approximating and solving the MDP:** The first step in each iteration of the main loop is to retrieve the current state from the target environment (line 3). Afterwards, an empirical transition model $P_{target}(s,a,s')$, is constructed from the current MENID rules (line 4). This construction is detailed in Sec. II-D. $P_{target}$ is used to find the action best suited for the current state, solving an MDP (line 5). Our algorithm does not impose any particular solver. Some options are standard algorithms like Value Iteration or Thompson sampling [13]. There is also state-of-the-art solvers like PROST [14], Gourmand [15], or FF-Hindsight [16].

**Testing an action:** Low confidence on the empirical outcome distribution inside the test environment is detrimental to the calculation of $P_{target}(s,a,s')$ (Sec. II-D). We determine that this is the case when $\delta$, the estimated error of the empirical distribution (Sec. II-C), is larger than a predefined threshold, e.g. 0.01 (line 6). Action testing is time-limited (lines 9, 11 and 13). Each experience is extracted and added to $E$, labeled as *test* (lines 10 and 12). The algorithm marks tested actions (line 7), so they are not tested again in the next iteration. This avoids an overly cautious behavior.

**Executing an action:** If an action is trusted enough, or if it has already been tested in the last iteration, it is scheduled for execution in the target environment. This action is unmarked so it is eligible again for testing (line 16). The action is executed (line 17) and the experience is added to $E$ labeled as *target* (line 18).

**Updating MENID rules:** At the end of the iteration, MENID rules are updated according to the new experiences appended to $E$ (line 20).

**Algorithm 2** $\delta$ bound calculation

**Input:** confidence $\varepsilon$, vector $\boldsymbol{\alpha} = [1 + x_0 \ldots 1 + x_n]$, sample size $S$

**Output:** $\delta$ s.t. $Pr(|p_i - q_i| > \delta, 0 \leq i \leq n) \leq \varepsilon$

1: $D \leftarrow \mathrm{Dir}(\boldsymbol{\alpha})$
2: Initialize vector $\boldsymbol{e} = [e_1 \ldots e_S]$
3: **for** $1 \leq j \leq S$ **do**
4: $\boldsymbol{p'} = [p'_1 \ldots p'_n] \leftarrow$ *sample*$(D)$
5: $e_j \leftarrow \max_i |p'_i - \frac{x_i}{\sum_{i'=0}^{n} x_{i'}}|$
6: **end for**
7: Sort $\boldsymbol{e}$
8: $q \leftarrow$ *round*$((1-\varepsilon) \cdot S)$
9: **return** $e_q$

### C. Estimating a Categorical Distribution

Algorithm 1 needs to evaluate whether the distribution of an action in the test environment is estimated with enough confidence (line 6). Should that be the case, further testing can be skipped. Sec. II-A already described the simplest way to estimate the distribution's parameters. However, we also need to evaluate the quality of the estimation.

Thompson [17] gives a method to find sample size so

$$Pr(|{}^{j}p_{r,i} - {}^{j}q_{r,i}| > \delta, 0 \leq i \leq n_r) \leq \varepsilon, \tag{3}$$

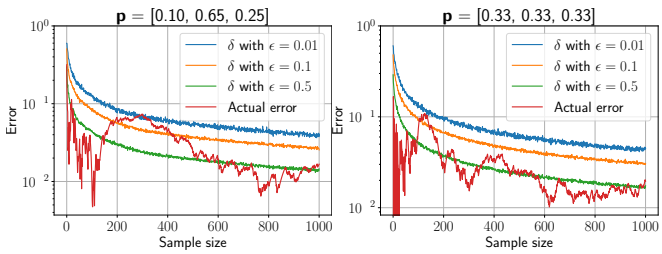
where ${}^{j}q_{r,i}$ is an estimation of $p_{r,i}$, $\delta$ is the maximum error desired for the parameters of the categorical distribution, and $1-\varepsilon$ is the confidence on having the estimation error within $\delta$ tolerance.

For each $\delta, \varepsilon$ and number of outcomes $(n_r + 1)$, Thompson's approach calculates the required sample size by approximating the estimation of each parameter by a Gaussian distribution. It assumes the worst-case scenario ${}^{j}p_{r,i} = \frac{1}{1+n_r}$. This results in conservative estimates of the sample size.

We work the other way around: given the vector of frequencies $\boldsymbol{f} = [{}^{j}x_{r,0} \ldots {}^{j}x_{r,n_r}]$ and the desired $\varepsilon$, we calculate a $\delta$ bound. We stop drawing observations when $\delta$ gets below a certain threshold.

Obtaining $\delta$ analytically is difficult. However, it can be easily approximated with the help of sampling. It is known that the conjugate prior of the categorical distribution is the *Dirichlet distribution*, $\mathrm{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = [1 + {}^{j}x_{r,0} \ldots 1 + {}^{j}x_{r,n_r}]$. This assigns a likelihood to each choice of ${}^{j}p_{r,i}$ parameters in base to the observations. The particular case of $n_r = 1$ (the noise effect and a regular outcome) corresponds to the Beta distribution.

The full process for approximating $\delta$ is described in Algorithm 2. A Dirichlet distribution $D$ over the parameters is defined (lines 1). Then, $D$ is sampled $S$ times (lines 3, 4). For each sampled set of parameters, the error with respect to $D$'s mode is computed and stored in the $\boldsymbol{e}$ vector (line 7). Finally, the $q$th error in the sorted $\boldsymbol{e}$ vector is picked, where $q$ is the index that corresponds to quartile $(1-\varepsilon) \cdot S$ (lines 9, 10, 11). If $S$ is sufficiently high, $e_q$ is a very accurate approximation to $\delta$. Fig. 2 shows two examples in which this is the case.




(a) (b)

Fig. 2. Error bounds ($\delta$) for the estimation of two categorical distributions as sample size increases. The actual error is calculated as $\max_i |{}^j p_{r,i} - {}^j q_{r,i}|$. Notice that, in both (a) and (b), the bounds follows quite closely the trend of the error. Moreover, the bounds with $\varepsilon = 0.01$ and $\varepsilon = 0.1$ are seldom exceeded. The worst case scenario analyzed by Thompson (uniform distribution) is shown in (b).

### D. Combining Information from Multiple Sources

Experiences come either from the test or the target environment. We need a strategy to merge these experiences. Let us first present two basic results:

*Proposition 1:* If the test environment is a perfect representative of the target one, then ${}^1p_{r,i} = {}^2p_{r,i}$, and ${}^1p_{r,i}$ can be estimated taking full advantage of the test sample frequencies:

$$ {}^1p_{r,i} \approx {}^1q_{r,i} = \frac{{}^1x_{r,i} + {}^2x_{r,i}}{N_1 + N_2}, \tag{4} $$

where $N_j = \sum_{i=0}^{n_r} {}^jx_{r,i}$.

*Proof:* $\boldsymbol{f} = [{}^jx_{r,0} \dots {}^jx_{r,n_r}]$ is drawn from a multinomial distribution with parameters $N_j, [{}^jp_{r,0} \dots {}^jp_{r,n_r}]$. Therefore, $\mathbb{E}\{{}^jx_{r,i}\} = {}^jp_{r,i} \cdot N_j$. Since ${}^1p_{r,i} = {}^2p_{r,i}$, $\mathbb{E}\{{}^jx_{r,i}\} = {}^1p_{r,i} \cdot N_j$. This means that $\mathbb{E}\{{}^1q_{r,i}\} = \frac{\mathbb{E}\{{}^1x_{r,i}\} + \mathbb{E}\{{}^2x_{r,i}\}}{N_1+N_2} = \frac{{}^1p_{r,i} \cdot (N_1+N_2)}{N_1+N_2} = {}^1p_{r,i}$. ■

In general, however, the distributions of the environments are different, and the following proposition applies.

*Proposition 2:* If there is any difference between the two distributions, the best estimator for ${}^1p_{r,i}$ considers only the target, since including the frequencies from an alien population would bias the estimator.

*Proof:* The same reasoning as before can be used to see that, when ${}^1p_{r,i} \neq {}^2p_{r,i}$, $\mathbb{E}\{{}^1q_{r,i}\} = \frac{N_1}{N_1+N_2} \cdot {}^1p_{r,i} + \frac{N_2}{N_1+N_2} \cdot {}^2p_{r,i}$. Therefore, by the law of large numbers, the best estimator of ${}^1p_{r,i}$ when $N_1 \to \infty$ uses only samples from $j = 1$. ■

However, when $N_1$ is small (as it is initially), and assuming that the test environment distribution is reasonably close to the target one, the outcome frequencies in the test environment may help establish a prior for the target probabilities. If the test frequencies are given a weight that decreases with $N_1$, we can ensure that, in the limit, our estimation will be unbiased. This same principle motivates the *decreasing-m-estimate* [18] that we reformulate for our purposes:

$$ {}^1q_{r,i} = \frac{{}^1x_{r,i} + \frac{m}{\sqrt{1+N_1}} {}^2x_{r,i}}{N_1 + \frac{m}{\sqrt{1+N_1}} N_2}, \tag{5} $$

where $m$ is the weight given initially to the test executions.

When $N_1 \to 0$, the estimation will be based almost entirely on the test executions. Conversely, when $N_1$ increases, the weight for the test executions decreases, basing the estimation more and more on the target executions.

Line 20 of Algorithm 1 updates the ${}^1p_{r,i}$ probabilities of the MENID rules according to the decreasing-m-estimate, while the test environment rules are updated with the simple estimate described in Sec. II-A. The most updated probabilities are always available for planning when constructing the transition model $P_{target}(s,a,s')$ in line 4. Specifically, $P_{target}(s,a,s') = {}^1q_{r,i}$ if rule $r \in a$ triggers in $s$, and $s \xrightarrow[\Omega_{r,i}]{} s'$ (i.e. $s'$ follows after the $i$th outcome). If the transition $s \xrightarrow[a]{} s'$ is impossible, $P_{target}(s,a,s') = 0$.

## III. Simulating Robotic Actions

In order to handle accurate simulations at a near-interactive computation time, we chose SOFA Framework [11]. SOFA offers the possibility to handle accurate simulations of rigid and deformable objects at interactive time performance. Therefore, it represents a good alternative to more traditional mechanical simulators that do not handle the object dynamics at a reasonable framerate, as well as game engines that do not handle accurate physics. As it is an open-source software, we were also able to design the scenarios as well as controlling the parameters as much as we wanted.

### A. Simulated Actions

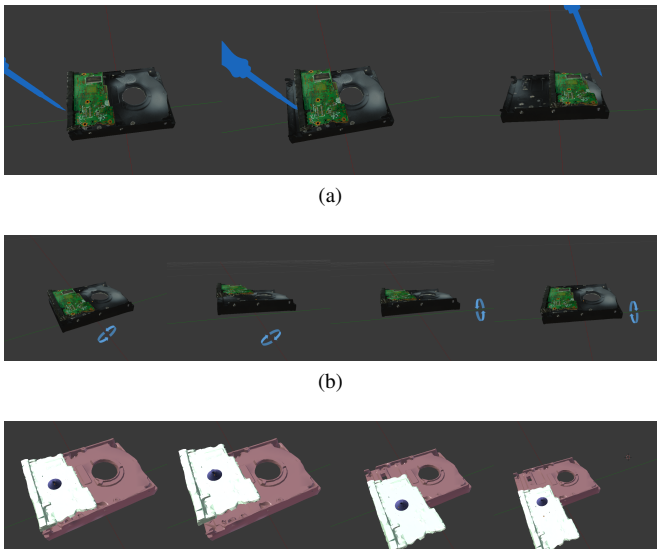
Within the use case of dismantling electro-mechanical devices, we chose three different robotic actions meant for disassembling a hard drive. The three actions are: (1) levering the Printed Circuit Board (PCB) or from the bay; (2) shaking the bay to remove the PCB; and (3) sucking the PCB with a suction cup and removing it from the bay. We enrich actions with parameters that give nuance over their physical realization and that could influence their success probability (e.g. discrete values of direction and force for *lever*).

These three actions are subject to non-determinism in the action execution, as manipulation skills depend on factors that escape our control in the real environment (e.g. friction, jerky motions, errors in perception). Simulating these actions require the use of a simulator able to handle the simulations of complex scenes generated from real data.

The resolution of the meshes is a key component for obtaining accurate results, at the price of a high computation time. Therefore, we present in Sec. IV-A a study evaluating the impact of the model quality on the symbolic error.

### B. The Resolution Problem

One of the first concerns that arise when using simulation as a proxy environment is realism and performance. On the one hand, we would like the simulator to behave as close as possible as the real world. For that, we need to include detailed 3D meshes that model the real life component. On the other hand, we want the simulator to be efficient enough so it can execute a large number of simulations, and detailed mesh models make collision detection more complex.



(a)

(b)

(c)

Fig. 3. Illustration of the three scenarios: (a) First scenario: the lever (blue) is used to remove the PCB (green) from the bay; (b) Second scenario: different shaking angles are applied to the bay to remove the PCB from it; (3) Third scenario: the PCB (white) is removed from the bay (red) using a suction cup (in black, located on the PCB).

At our disposal we have a library of 3D scans of hard drive components that we use to test our approach. We have full control on the detail of such models. We deem sensible to measure the impact of the downgrade. We perform a test on the accuracy and efficiency of the simulated actions with respect to the most detailed model (Sec. IV-A).

## IV. Experimental Study

First, we show that simulations with models of different quality yield substantially different qualitative results. On the one hand, this imposes the need to compromise performance to accuracy and vice versa. On the other hand, this shows that simulations with different qualities could already be considered different environments.

Second, we provide evidence that Algorithm 1 takes advantage of fast simulations to quickly learn the outcomes probabilities and to exploit the best actions in the target environment. We show this by plugging in a high-quality SOFA simulation as target environment.

### A. The Mesh Resolution Problem

We have evaluated the performance/accuracy trade-off of the actions in three scenarios[1]

*1) First scenario: levering a PCB:* The first scenario considers a rigid lever that is used to remove a PCB from the bay of the hard disk (see Fig. 3a). All the objects are rigid. We used an Euler implicit integration scheme. Objects' masses were set to their real world counterpart's. In this task, a good location on the PCB to perform the levering action must be found. In our simulations, we evaluated 20 different positions of the lever, located at the boundary of the PBC.

[1]The computer specifications for these experiments are: Linux Kernel 5.2.11-100 x86_64 Fedora 29, Intel(R) Xeon(R) CPU E5-1603 v4 @ 2.80GHz, 16GiB Ram, GeForce GTX 1080.

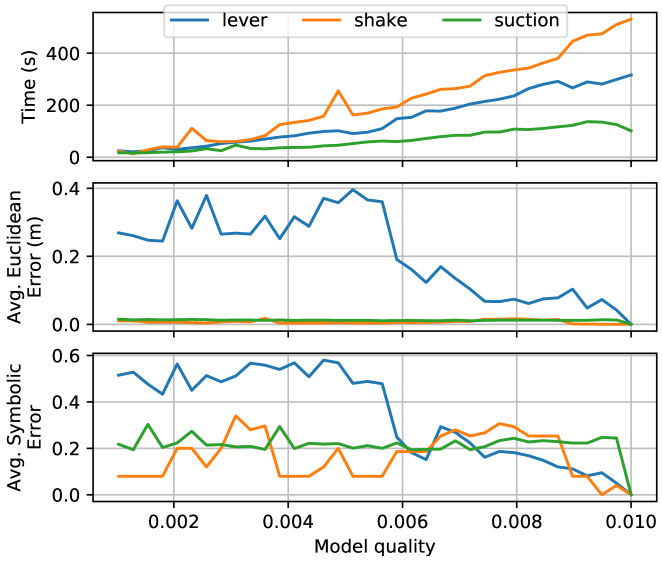



Fig. 4. Different performance measures for the *lever*, *shake* and *suck* action as the quality of the models (i.e. proportion of retained vertices) varies.

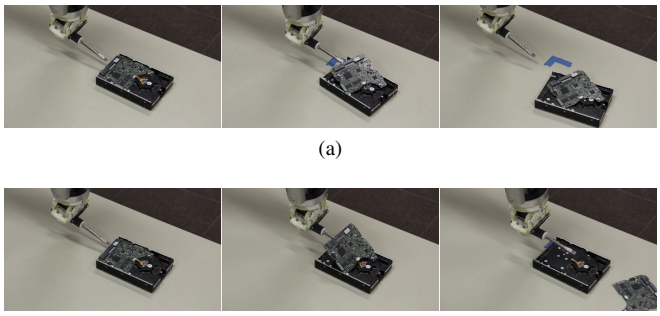
*2) Second scenario: shaking the bay:* The second scenario consists in shaking the bay to remove the PCB from it (see Fig. 3b). We used the same simulator configuration as for the first scenario. In this task, we tested for 10 different angles of shaking movement, ranging from 0.01 to 5 degrees.

*3) Third scenario: sucking the PCB:* In the third scenario, we used a deformable suction cup to suck the PCB and remove it from the bay (see Fig. 3c). We used the method proposed by Bernardin *et al.* [19] in this scenario. Both the PCB and the bay were rigid objects with the same properties as for the two first scenarios, while the suction cup was modeled as a deformable object using the co-rotational Finite Element Method (FEM). The success of the task is highly dependent on the location of the suction cups as well as the pressure applied on the PCB. In our simulations, we tested for 9 different positions spread on a $3 \times 3$ grid on the PCB. We also applied 3 different suctions pressures, ranging from -0.001 to -1135 Pa (difference with atmosphere pressure, 101135 Pa).

The mesh quality was defined as the ratio of the number of points between the reduced models and the original one. All experiments were performed on 36 different mesh qualities, ranging from 0.00103 to 0.01 with 5 repetitions for each set of parameters. For the lowest quality, the PCB had 303 vertices and 762 triangles while the bay had 491 vertices and 1082 triangles. For the highest quality, the PCB had 3310 vertices and 6776 triangles while the bay had 5183 vertices and 10466 triangles.

Results are shown in Fig. 4. The symbolic error is measured as the $1 - J(s, s')$, where $s, s'$ are the collection of predicates that represent the state at the end of a low quality simulation and its high quality counterpart, respectively; and $J(s, s')$ is the Jaccard index between both sets:

$$J(s, s') = \frac{|s \cap s'|}{|s \cup s'|} \tag{6}$$



(a)

(b)

Fig. 5. Two different lever motions for levering and removing a PCB from the hard drive reproduced with the robot arm. The first motion, (a), is unsuccessful, while the second, (b), is successful.

Pikes on the graphs can be explained as noise on the initial conditions of the experiment. The irregularities of the models may result in the objects being displaced.

The plot already allows us to identify the trend that we anticipated in the lever action, with an inflection point around 0.006: higher qualities result in more execution time and in results significantly different from lower qualities.

### B. Interleaved Learning and Planning

Our learning algorithm has been evaluated using two configurations of SOFA and a real-life robot setting as the available environments. The scenes used in our evaluation contain a variety of hard-drive scenarios. The goal in each scenario is the removal of the PCB by means of the available actions (motions with different application points).

One simulation environment has low quality ($q = 0.001$) models. While simulations in this high-performance setting run promptly, the accuracy of the results varies sensibly with respect to the high-quality models, as depicted in Fig. 4. The other simulation has high-quality models ($q = 0.01$). This setting takes, on average, much more time than its low-resolution counterpart. It can be appreciated that the outcomes vary sensibly, which suggests that high-quality models capture more complex interactions. To account for stochasticity, random noise is added to the input trajectories and the physical parameters (friction and restitution).

The robot environment features a WAM robotic arm in front of a table. On the table is a hard drive with the PCB side facing upwards. Several levering motions can be executed to pull the PCB out of the drive, and some of these actions are more successful than others, as shown in Fig. 5.

Our baseline learns purely from the target environment. This can be achieved simply by setting $T = 0$ in our algorithm (this effectively disables the execution of actions in the test environment). We compare this to $T = 20$ (i.e. 20 seconds for testing an action). We use the *m-estimate* described in Sec. II-C with $m = 10$. The success or failure of an action is rewarded and penalized, respectively (benefit vs risk tradeoff). To evaluate, we use the accumulated reward (score) over several episodes for a total duration of 1 hour.

We have run several sets of experiments assigning different rewards to successful and unsuccessful executions of actions. In Fig. 6, we show in blue the result of trying to learn the best action(s) to execute in the high-quality simulation by interleaving executions of actions in the low-quality and the high-quality simulations. The red line shows what happens when only the target environment (the high-quality simulation) is available.

For each reward choice, we have run five experiments. The solver used to calculate the best action is a custom implementation of a Thompson sampler [13].

When there is no risk involved (i.e. the penalty for failing is 0), as in Fig. 6a, it is usually best to encourage experimentation in the target environment, since failing an action does not incur in any negative consequence and we sample results directly from the target distribution, while simulation only wastes time. However, when more relevance is given to failures, using the target environment for testing becomes too risky. Large negative reward can be accumulated, and the benefits of having a virtual environment to simulate actions and gain more information about them is more evident.

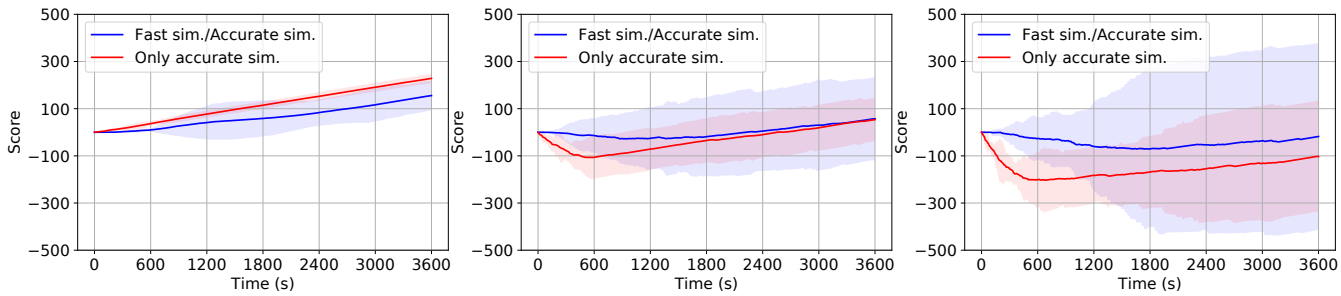
Our method works thanks to the following: (1) the test environment allows sampling the action outcomes from a probability distribution that is sufficiently similar to the target one; (2) this can be done in less time than using the target environment directly; and (3) risk-free testing avoids the occasional penalties and adverse effects that occur at the beginning, in the abscense of experience.

Fig. 7 shows the result for settings that involve the real-life robot. We show: (1) in blue, the low-quality simulation/real robot setting; (2) in red, the high-quality simulation/real robot setting; and (3) in green, what happens when only interactions with the real robot are available (baseline). These results show that, when simulation is involved, we avoid the accumulation of failures at the beginning of the risky environments (Fig. 7b, Fig. 7c). However, simulation requires time, so if no penalty, or mild penalties are associated to failures, direct executions with the robot or with less precise simulations give rewards much faster (Fig. 7a, Fig. 7b). Since our method will eventually decide to not simulate anymore, the end slope of the curves is always the same.

## V. Conclusions and Future Work

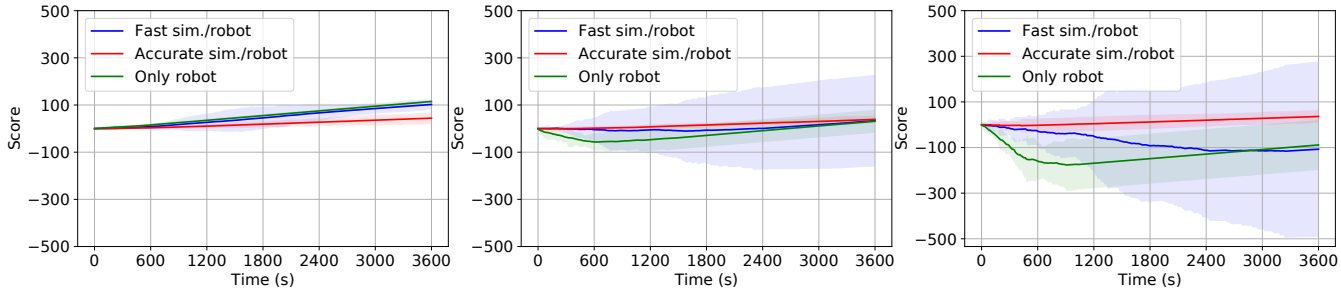
Motivated by the idea of using a simulator to gather symbolic information about the success ratio of robot actions, we have presented a framework that allows to learn from experiences performed in different environments. We have introduced MENID rules, a formalism to model different outcomes according to multiple environments and their uncertainty. We have also presented an algorithm able to decide in which environment execute the actions.

Our framework has been evaluated with scans of real objects and actions used in a hard drive disassembly task: lever, shake, and suck. The SOFA simulator has acted as test environment in a low-resolution configuration, and as target environment in a high-resolution one by adding noise. Experiments with a real robot have been also conducted. Compared to a standard strategy that learns only from




(a) (b) (c)

Fig. 6. Results when learning the accurate, but slow simulation environment (target environment). The reward for successful actions is always 1. In (a), there is no penalty for failing an action. In (b) and (c), failures are penalized with a decrement of 5 and 10 to the score, respectively. The blue line shows the accumulated reward when the fast but inaccurate environment is used as test environment. Red line represents a learner that can access exclusively the slow simulator. The shaded regions represent 3 standard deviations around the central trend.




(a) (b) (c)

Fig. 7. Results when the target environment is the real world (i.e. physical robot acting in a real-life setting). Scenario (a) features a risk-free environment (no penalty for failures), while scenarios (b) and (c) correspond to scenarios with a penalty of 5 and 10 for failed actions, respectively.

the target environment, our algorithm learns the actions' outcomes with less risk.

Once we have validated the learning of action outcomes, in the future we would like to extend the proposed algorithm to learn also new outcomes to the rules and even new rules from scratch. Finally, we would like to explore the case of more than two environments in detail.

## References


[1] M. L. Littman, "Probabilistic propositional planning: representations and complexity," in *Proceedings of the National Conference on Artificial Intelligence*, 1997.

[2] M. Eppe, P. D. H. Nguyen, and S. Wermter, "From semantics to execution: Integrating action planning with reinforcement learning for robotic causal problem-solving," *Frontiers in Robotics and AI*, vol. 6, p. 123, 2019.

[3] D. Aineto, S. Jiménez, and E. Onaindia, "Learning STRIPS Action Models with Classical Planning," *International Conference on Automatic Planning and Scheduling (ICAPS)*, 2018.

[4] Q. Yang, K. Wu, and Y. Jiang, "Learning action models from plan examples using weighted MAX-SAT," *Artificial Intelligence*, vol. 171, no. 2-3, pp. 107–143, 2007.

[5] S. N. Cresswell, T. L. McCluskey, and M. M. West, "Acquiring planning domain models using LOCM," *Knowledge Engineering Review*, vol. 28, no. 2, pp. 195–213, 2013.

[6] H. H. Zhuo, H. Muñoz-Avila, and Q. Yang, "Learning hierarchical task network domains from partially observed plan traces," *Artificial Intelligence*, vol. 212, no. 1, pp. 134–157, 2014.

[7] D. Draper, S. Hanks, and D. S. Weld, "Probabilistic Planning with Information Gathering and Contingent Execution," in *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, 1994.

[8] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, pp. 309–352, 2007.

[9] D. Martínez, G. Alenyà, and C. Torras, "Relational reinforcement learning with guided demonstrations," *Artificial Intelligence*, vol. 247, pp. 295 – 312, 2017, special Issue on AI and Robotics.

[10] N. Lipovetzky, M. Ramirez, and H. Geffner, "Classical Planning with Simulators: Results on the Atari Video Games," in *IJCAI*, 2015, pp. 1610 – 1610.

[11] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Yohan Payan, Ed. Springer, 2012, pp. 283–321.

[12] D. Martínez, G. Alenyà, T. Ribeiro, K. Inoue, and C. Torras, "Relational reinforcement learning for planning with exogenous effects," *Journal of Machine Learning Research*, vol. 18, no. 78, pp. 1–44, 2017.

[13] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on thompson sampling," *Found. Trends Mach. Learn.*, vol. 11, no. 1, p. 1–96, July 2018.

[14] T. Keller and P. Eyerich, "PROST: Probabilistic Planning Based on UCT," in *International Conference on Automatic Planning and Scheduling (ICAPS)*, 2012.

[15] A. Kolobov, Mausam, and D. S. Weld, "LRTDP vs. UCT for Online Probabilistic Planning," in *AAAI*, 2012, pp. 1786–1792.

[16] M. Issakkimuthua, A. Fern, R. Khardon, P. Tadepalli, and S. Xue, "Hindsight Optimization for Probabilistic Planning with Factored Actions," *International Conference on Automatic Planning and Scheduling (ICAPS)*, pp. 120–128, 2015.

[17] S. K. Thompson, "Sample size for estimating multinomial proportions," *American Statistician*, 1987.

[18] D. Martínez, G. Alenyà, and C. Torras, "Planning robot manipulation to clean planar surfaces," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 23–32, 2015.

[19] A. Bernardin, C. Duriez, and M. Marchal, "An interactive physically-based model for active suction phenomenon simulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1466–1471.