Autonomous Design of Experiments for Learning by Experimentation

Erwin Prassler, Björn Kahl, Timo Henne, Alex Juarez[1], Monica Reggianni[2]

In Artificial Intelligence, numerous learning paradigms have been developed over the past decades. In most cases of embodied and situated agents, the learning goal for the artificial agent is to „map“ or classify the environment and the objects therein [1, 2], in order to improve navigation or the execution of some other domain-specific task. Dynamic environments and changing tasks still pose a major challenge for robotic learning in real-world domains. In order to intelligently adapt its task strategies, the agent needs cognitive abilities to more deeply understand its environment and the effects of its actions. In order to approach this challenge within an open-ended learning loop, the XPERO project (`http://www.xpero.org`) explores the paradigm of *Learning by Experimentation* to increase the robot's conceptual world knowledge autonomously. In this setting, tasks which are selected by an action-selection mechanism are interrupted by a learning loop in those cases where the robot identifies learning as necessary for solving a task or for explaining observations. It is important to note that our approach targets *unsupervised* learning, since there is no oracle available to the agent, nor does it have access to a reward function providing direct feedback on the quality of its learned model, as e.g. in reinforcement learning approaches.

In the following sections we present our framework for integrating autonomous robotic experimentation into such a learning loop. In section 1 we explain the different modules for stimulation and design of experiments and their interaction. In section 2 we describe our implementation of these modules and how we applied them to a real world scenario to gather target-oriented data for learning conceptual knowledge. There we also indicate how the goal-oriented data generation enables machine learning algorithms to revise the failed prediction model.

1 A Framework for Design of Experiments

We propose a framework for designing experiments to be executed by a robotic learner which implements the paradigm of Learning by Experimentation. This framework integrates a *stimulation* and a *design of experiments* component which interact by using available knowledge about the environment. The design component consists of two parts which address the *exploration* of the feature space and the actual *experimentation*, which is the focus of our framework. It serves the purpose of designing and executing sequences of robot actions (*experiments*) in order to collect target-oriented data that afford learning new concepts. The output of the

1. Erwin Prassler, Björn Kahl, Timo Henne and Alex Juarez are with the b-it Applied Science Institute, Department of Computer Science, University of Applied Sciences Bonn-Rhein-Sieg.

2. Monica Reggianni is with the Dipartimento di Ingegneria e Gestione dei Sistemi Industriali, University of Padua, Italy and a visiting professor at the University of Applied Sciences Bonn-Rhein-Sieg.

experimentation module is thus intended to provide a machine learning algorithm with data in a format appropriate for learning conceptual knowledge.

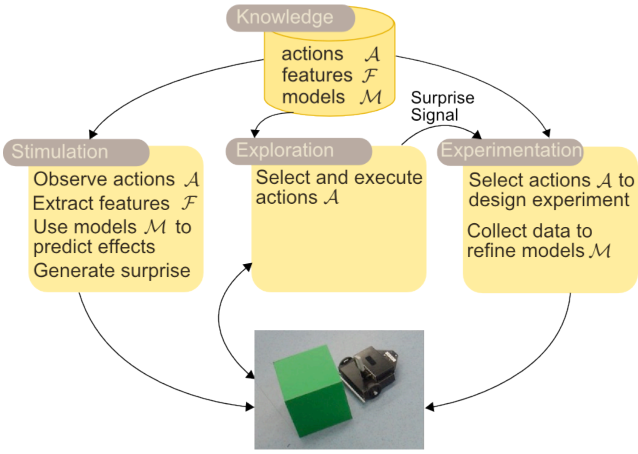


Figure 1. The proposed framework for autonomous design of experiments.

1.1 Available Knowledge

The components for stimulation and design of experiments rely on information available to the autonomous agent, that describes and relates robot actuation and sensing capabilities and knowledge about the environment. We define the *available knowledge* as the aggregate of the following information:

- *A,* the set of *actions* that the robot can execute. Each action $a_1 \in A$ is defined in a set of parameters $\theta_1, \theta_2, \ldots ,\theta_n$ which must be assigned values before the actual execution of the action.
- *F*, the set of *features* which are extracted from the robot's sensor data. These features range from direct sensory data such as odometry and bumper signals, over object characteristics (e.g. color, size, pose) to more complex constructs such as predicates in first-order logic, e.g. position(object,X,Y,time).
- *M*, the set of models of the worlds. These models represent the current beliefs of the agent on the effects that its actions have on the environment (e.g. „if a ball is hit with a force *f* at time *t*, it will move with velocity *v* in direction *d* until it comes to a stop at a time t_2 with $t_2 > t_1$").

Any of these parts of the agent knowledge, but especially the set of models *M*, are subject to revision and extension along the cognitive evolution process of the agent, facilitated by our proposed framework.

1.2 Design of Experiments

The process of discovering new concepts in robotics is still not well-defined in the state-of-the-art research literature. The existing solutions mostly focus on very specific domains. Consequently, no established general procedure exists which could be employed in determining the sequence of actions (experiment) that will successfully lead to an improvement of the agent's knowledge. According to [3] the process of knowledge abstraction should involve four steps: act, predict, surprise, and refine. The structure of our framework follows these steps that have been implemented in three modules: *exploration* (Sec. 1.3) *stimulation* (Sec. 1.4) and *experimentation* (Sec. 1.5).

The *exploration* module organizes how the agent uses its current knowledge to *act*, i.e. it selects and executes actions (A) to achieve predefined goals or to explore its environment. At the same time the *stimulation* module continuously observes the robot actions (A), extracts information about the environment as features (F) from sensor data and *predicts* the behaviour of these features by using the current knowledge, represented by models M. When an unexpected phenomenon (a surprise) is observed, a signal is sent to the *experimentation* block. The *experimentation* module collects information about the experienced surprise through the selection of action sequences (experiment). These sequences are designed to provide relevant data to the learning module, thus starting a process of *revising* and *refining* the current model M in order to improve its predictive capabilities.

1.3 Investigating the Environment: Exploration

The task of the *exploration* module is to identify which paradigm will provide relevant information from the environment. As experimental paradigm we define a distinguishable class of experimental situations, i.e. distinctively different ways in which the agent investigates the environment [4]. The initial set of experimental paradigms (P) is built from the set of elementary actions A that the robot can execute. In later stages, the autonomous agent will try to produce new experimental paradigms, e.g. by combining known paradigms [4], also taking into account the cost and complexity of their execution.

Choosing the most suitable paradigm from P and the combination of its elements is a difficult task. One solution lies in applying a heuristic to choose an appropriate paradigm, taking into account the current knowledge, the costs of the experimental paradigms, and the exploration goal(s).

In our framework we introduced three initial heuristics suggested by [5]. One heuristic ($H_{goalSeeking}$) chooses an experimental paradigm known to change a feature in F with the objective of modifying its value with a certain relation to a target value. A second heuristic ($H_{noEffect}$) explores the paradigms that apparently have no effect on the environment. This heuristic aims at validating current beliefs on these paradigms, and tries to produce effects which had not been encountered previously. Finally, the heuristic (H_{random}) explores a randomly selected paradigm with randomly defined parameters. By applying these heuristics, we can guarantee that after a reasonable execution time, the system will have investigated even the paradigms which are not so promising, but that could still contribute to the creation of new models M.

1.4 From Exploration to Experimentation: Stimulation

A central question within Learning by Experimentation is when to stop exploring the environment heuristically, and start the design and execution of Experiments. We believe that in order to facilitate autonomous, open-ended learning, the trigger of the experimentation phase should be intrinsic, automatic, and at the same time related to the robot's experience during the exploration. In this work we propose the use of a robotic surprise mechanism to stimulate the design of experiments.

The application of artificial surprise in various fields such as evolutionary and developmental robotics, social agents, and human-machine interaction have shown the effectiveness and scalability of employing this concept. In the literature we can find examples of the integration of artificial surprise to active vision and adaptive learning [6, 7, 8, 9], as well as approaches to robot exploration of a partially unknown environment [10, 11] and [12, 13]. These approaches share the idea that surprise is the result of an observation that diverges from an expectation.

The surprise mechanism used in this paper combines several elements from the mentioned approaches to artificial surprise and works under the assumption that the knowledge available to the robot can *predict* and *explain* any observation derived from the effect of the robot actions on the environment. To achieve this, each action $a_i \in A$ is associated with one or more models $m_i \in M$. If an action brings about an *observation* that diverges from the prediction offered by the associated model, this is considered as surprise.

The robot recognizes events that are candidate to surprise on two different levels of abstraction. The first level is directly related to the sensory input data and simulates a reflex to these events. At this level, the model is an estimation of the underlying probability distribution of the sensor data where such distribution is updated periodically as the robot executes its actions. The second level of abstraction uses available knowledge represented as *first-order logic* models or *qualitative* models to attempt an explanation of physical phenomena associated to the execution of an action, for example the rolling of a ball after the robot has pushed it.

Before the execution of an action, the models predicting its effects are loaded into memory. During execution, the sensor data is converted into the corresponding representation and compared online with these models. If the observation shows a divergence from the expected effect, a signal indicating a prediction failure is produced. This surprise can be characterized as a disconfirmed active expectation.

1.5 Experimentation

The *experimentation* module receives a surprise signal from the *stimulation* module whenever an observation diverges from the prediction. This signal contains information about the initial state of the environment as perceived by the robot, the experimental paradigm and the parameter values which generated the surprise, and the prediction rule which failed.

The agent must identify those features of the initial states and those distinctive parameters of the experimental paradigms which were relevant to the prediction failure, in order to avoid storing too much redundant or irrelevant data. Early attempts to form equivalence classes in collected data can be found in [14] using k-means clustering algorithm and support vector machines (SVM) to define affordance relations [15]. While these approaches attempt to directly identify the final relations that will be part of robot knowledge, our goal here is instead to provide a heuristic that can drive the agent in the experimentation phase.

The correct identification of the initial situation and paradigm can reduce the search space for learning algorithms significantly, which is critical for the task of learning high-level concepts, such as models in first-order logic. Errors in this identification process will most probably result in an ineffective learning phase, however the overall correctness of the framework is not affected.

An additional improvement of the framework, and part of our future work, is a mechanism to define the importance of the stored surprises. As suggested in [16], exploration may be achieved by selecting actions and/or states which have been selected less frequently or less recently. The importance of a surprise can therefore be inversely related either to its age or to its recency, i.e. the time during which a surprise did not occur.

2 Framework Implementation

Starting from the description presented in Section 1 we developed a first implementation of the proposed framework. The system was designed to be easily adaptable to new experimental setups. We built a library based on the middleware ICE [21] with immutable parts (the framework algorithms) and clearly defined the interfaces where the description of the available knowledge and the interface with the real robot sensors and actuators can be provided, depending on the current setup. Thus, different robots and sensor types can be integrated without affecting the overall framework.

2.1 Application to a real scenario

To validate the framework we used the library described above to support the collection of data within the showcase outlined in the XPERO project. This showcase features a robot located in an almost empty room with boxes blocking the room's exit. Although for a human programmer the solution to this task is straightforward, this scenario still presents a major challenge for the currently available unsupervised learning paradigms.

In this work, we focused on the first two concepts that the agent should learn in the XPERO evolution of theories. For this the robot, an educational embodiment named Eddy (see [17]), is situated in a free space with static and movable objects (Figure 2). Additionally, an overhead camera provided the object IDs and localisation information about the robot and the objects. The notions which can be learned in these subscenarios include the notion of *static* vs. *movable* objects and the notion of *obstacles*, i.e. that under certain circumstances objects prevent a robot from moving to certain places or in certain directions.

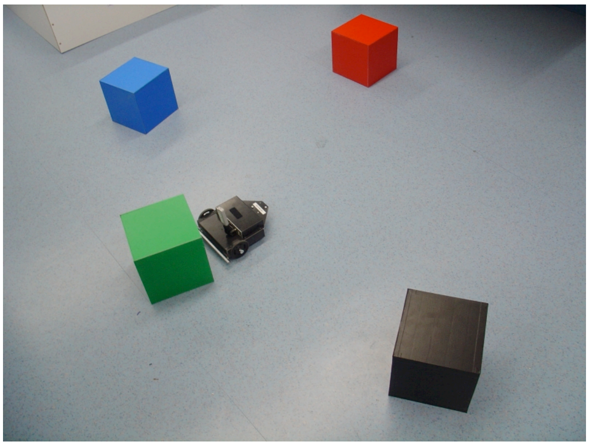


Figure 2. The real environment used in our experiments

Four elementary actions were implemented to define the initial set of paradigms P. Tables 1 and 2 present these actions and their parameters, respectively. The predictive rules in M available to the robot were encoded in first order logic. For the subscenarios two of them were used, given below in their Prolog notations and their associated actions:

a_2: move(Object,Start,Dist,End):- approxEqual(Start,End).
a_3: move(Robot,Start,Dist,End):- approxAdd (Start,Dist,End).

Their meaning is straightforward:

1. When the robot executes a_2 on a certain object, its model predicts that the end position of the object will be approximately equal to its initial position. This knowledge cannot explain cases when the robot tries to push objects that can actually be moved, which generates a prediction failure.
2. For the execution of action a_3, the model predicts that the robot end position will be approximately equal to the sum of the robot start position and the distance parameter of the action. This prediction fails for cases when the robot bumps into non-movable objects on its path.

	ActionId	Parameters
a_1	goInContact	objectId
a_2	pushObject	objectId, pushDistance
a_3	moveForward	distance
a_4	rotate	angle
a_5	goTo	xCoord, yCoord, orientation

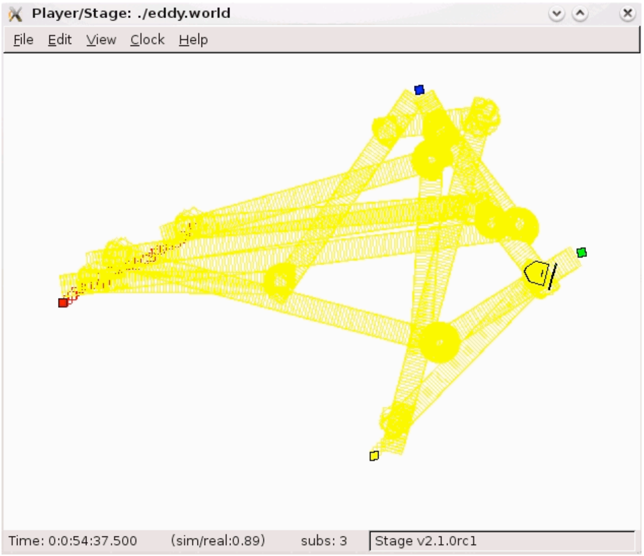
Table 1. Actions *A*

Parameter	Domain
objectId	objects in the environment
pushDistance	[minPushDist, maxPushDist]
distance	[minDist, maxDist]
angle	[minAngle, maxAngle]
xCoord	[-10.0, 10.0]
yCoord	[-10.0, 10.0]
orientation	[-π, π]

Table 2. Parameters *P*

Upon receiving a surprise signal, the experimentation module autonomously designs an experiment by selecting an appropriate paradigm, defining the initial states for the environment, and choosing the paradigm parameters that efficiently cover the experimental domain as shown in [18]. Once the experiment has been designed, a planner produces the sequence of actions that the robot will perform to execute the experiment.

As a simple example, consider the case where the robot executes action a_2 (pushObject) while interacting with a movable object, encountering a surprise and triggering the design of experiments. The framework was able to correctly identify the action generating the surprise and to design a new experiment to explore the action parameters, i.e. the object in the environment and the distance for the pushing actions, each time starting from a new robot pose.



(a)

```
goTo 0.217194 -0.894004 -2.95998
pushObject( 3, 0.351592 )
goTo 0.729848 0.347354 -0.564906
pushObject( 2, 0.540063 )
goTo 0.988094 0.214923 2.82446
pushObject( 1, 0.387043 )
goTo -1.31636 -0.310019 -1.61959
pushObject( 2, 0.122563 )
goTo 0.612992 1.32164 1.24174
pushObject( 4, 0.231988 )
goTo 1.22754 -0.198222 -0.821006
...
```

(b)

Figure 3. An execution trace in a simulated environment (a) that is the result of the execution of the experimental plan (b).

An experimental trace logged during an experimentation phase is shown in Figure 3(b). The execution of the experiments gathers the data necessary to learn a new model, which is able to correctly explain the observations made by the robot when trying to push an object, regardless of it being movable or not. Figure 3(a) depicts the simulated execution path of the plan covering several experiments by a robot in the environment previously described.

As intended by our framework design, we provided the specifically targeted data generated in our experiments in a real environment to HYPER, a machine learning tool for inductive logic programming (ILP) [19]. In order to discover the desired concepts in the form of predicates, HYPER was extended to facilitate predicate invention. However, when dealing with data generated by unspecific robot actions, HYPER is not able to derive any concept, since the amount of data and possibly significant variables inevitably led to combinatorial explosion problems. Here our framework proved as a useful bias for revising the prediction model under question, since it focusses on generating data for the action whose prediction model failed and produces data in a predicate form, thus limiting the number of variables to be investigated by the ILP algorithm. With these data, HYPER was able to learn the concepts of movable objects and obstacles [20], which could not be achieved with data from unspecific robot actions.

3 Conclusion and Future Work

We have presented ongoing work on a framework for integrating targeted data generation for robotic Learning by Experimentation. We explained how the different modules stimulation, exploration and experimentation work together to enable an intrinsically motivated, reasonable and autonomous switch from task execution to experimentation. Subsequently we showed how the data collection in the experimentation phase is guided by the heuristic applied in the exploration phase, and by the robotic surprise from the stimulation module. We descibed how the implemented framework library allows for a simple exchange of robots, sensors and scenarios. By applying our framework to a real world scenario, we were able to show its feasibility and demonstrate how purposeful data generation takes place which enables a learning algorithm to discover conceptual knowledge.

Our current work focusses on exploring other learning algorithms and evaluate the effect of both the quality of the experiment design, and the number of the experiments performed, on the prediction accuracy of the revised model. Furthermore, we are further developing the automation of generating and evaluating experimental paradigms, and exploring other knowledge representations such as qualitative models, and test our framework with different scenarios.

References

[1] D. F. Wolf and G. S. Sukhatme, 'Mobile robot simultaneous localization and mapping in dynamic environments', Autonomous Robots, 19(1), 53–65, (2005).
[2] S. Petti and T. Fraichard, 'Safe motion planning in dynamic environments', Intelligent Robots and Systems (IROS). August, 2005.
[3] W.-M. Shen, 'The process of discovery', Foundations of Science, 1(2), 233–251, (1995).
[4] P. C.-H. Cheng, 'Modelling experiments in scientific discovery', in International Joint Conferences on Artificial Intelligence, IJCAI, pp. 739–745, (1991).
[5] W.-M. Shen, 'Discovery as autonomous learning from the environment', Machine Learning, 12(1), 143–165, (1993).
[6] M. Peters, 'Towards artificial forms of intelligence, creativity, and surprise.', in Proceedings of the 20 th Meeting of the Cognitive Science Society, pp. 836–841, (1998).
[7] M. Peters and A. Sowmya, 'Active vision and adaptive learning', in Proceedings of the 15th. Conference on Intelligent Robots and Computer Vision, volume 2904, pp. 413–424, (1996).
[8] L. Itti and P. Baldi, 'A principled approach to detecting surprising events in video', in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 631–637, San Siego, CA, (June 2005).
[9] L. Itti and P. Baldi, 'Bayesian surprise attracts human attention', in Advances in Neural Information Processing Systems, Vol. 19 (NIPS*2005), pp. 1–8, Cambridge, MA, (2006). MIT Press.
[10] L. Macedo and A. Cardoso, 'Exploration of unknown environments with motivational agents', in Third International Joint Conference on Autonomous Agents and Multiagent Systems, (2004).
[11] L. Macedo, A. Cardoso, and R. Reisenzein, 'A surprise-based agent architecture', in Cybernetics and Systems, ed., R. Trappl, volume 2. Austrian Society for Cybernetics Studies, (2006).
[12] F. Kaplan and P-Y. Oudeyer, 'Curiosity-driven development.', in Proceedings of the International Workshop on Synergistic Intelligence Dynamics, (2006).
[13] P-Y. Oudeyer, F. Kaplan, and V. Hafner, 'Intrinsic motivation systems for autonomous mental development', IEEE Transactions on Evolutionary Computation, 11(2), 265–286, (2007).
[14] M. R. Dogar, M. Cakmak, E. Ugur, and E. Sahin, 'From primitive behaviors to goal-directed behavior using affordances', in IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems, (2007).
[15] J. Mugan and B. Kuipers, 'Learning distinctions and rules in a continuous world through active exploration', in 7th International Conference on Epigenetic Robotics, (2007).
[16] S. Thrun, 'The role of exploration in learning control', in Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches, eds., D.A. White and D.A. Sofge, Van Nostrand Reinhold, Florence, Kentucky 41022, (1992).
[17] L. Bertelli, F. Bovo, L. Grespan, S. Galvan, and P. Fiorini, 'Eddy: an open hardware robot for education', in 4th International Symposium on Autonomous Minirobots for Research and Edutainment (AMIRE), Buenos Aires, Argentina, (October 2007).
[18] F. Di Palma, M. Reggiani, and P. Fiorini, 'Design of experiment for qualitative equation discovery: a comparison', in Eurosim Congress on Modeling and Simulation, Ljubljana, Slovenia, (September 2007).
[19] I. Bratko, Prolog Programming for Artificial Intelligence, Addison Wesley Publishing Company, 2001.
[20] G. Leban and I. Bratko, 'Discovering notions using hyper', Technical report, University of Ljubljana, Artificial Intelligence Laboratory, (February 2008).
[21] M. Henning, 'A new approach to object-oriented middleware', IEEE Internet Computing, 8(1), 66–75, (2004)