

Surprise-Based Learning for Developmental Robotics

Nadeesha Ranasinghe and Wei-Min Shen
 Information Sciences Institute,
 University of Southern California
 E-mail: nadeesha@isi.edu, shen@isi.edu

Abstract

This paper presents a learning algorithm called surprise-based learning (SBL) capable of providing a physical robot the ability to autonomously learn and plan in an unknown environment without any prior knowledge of its actions or their impact on the environment. This is achieved by creating a model of the environment using prediction rules. A prediction rule describes the observations of the environment prior to the execution of an action and the forecasted or predicted observation of the environment after the action. The algorithm learns by investigating “surprises”, which are inconsistencies between the predictions and observed outcome. SBL has been successfully demonstrated on a modular robot learning and navigating in a small static environment.

1. Introduction

Cognitive development on a robotics platform is a very ambitious research task for many fields, including robotics, machine learning, cognitive science, developmental psychology and many others [1-4]. In this paper, we envision a scenario of learning as follows. First a “baby” robot will be arbitrarily placed in a new, small and static environment and it will autonomously perform learning and problem solving with a limited number of sensors and actions. Then, we will vary the number of sensors and actions to determine the competence of the learning. We will even dynamically “sabotage” the sensors (such as injecting noise or turning a sensor upside-down) and test if the learning can adapt to new situations. This is similar to testing a human’s learning ability by having them wear glasses that invert their vision.

In machine learning, related approaches include those algorithms that are supervised learning (SL), unsupervised learning (UL) or based on reinforcement (RL) [5]. However, an external supervisor may not always be present in the situation here. Unsupervised learning intends to ignore most feedback from the environment that may be critical for developmental learning. Reinforcement learning receives feedback from the environment. Such approaches may include Evolutionary Robotics [6] and Intrinsically Motivated Reinforcement Learning [7] that use physical robots. However, most of these algorithms focus on learning a policy from a given discrete state model (or a world model) and the reward is typically associated with a single goal. This makes transferring the learned knowledge to other problems more difficult.

Other approaches that are more related to the tasks here include Complementary Discrimination Learning (CDL) [8] that attempts to learn a world model from a continuous state

space and is capable of predicting future states based on the current states and actions. This facilitates knowledge transfer between goals and discovering new terms [9]. However, CDL is more logical-based learning and has not been applied to physical robots to learn directly from the physical world. In addition, CDL only performs generalization and specialization on a complementary rule, while other essential activities, such as abstraction, surprise analysis with noisy sensors, dynamic goal assignment, prediction creation and continuous maintenance, are not yet properly addressed.

Evolutionary Robotics (ER) is a powerful learning framework which facilitates the generation of robot controllers automatically using neural networks and genetic programming. The emphasis in ER is to learn a controller given some model or detailed information about the environment, which may not be readily available. However, advances in ER such as the Exploration-Estimation Algorithm [10] can learn an internal model for actions or sensors, but often need to reset the environment and the robot’s position.

Intrinsically Motivated Reinforcement Learning (IMRL) uses a self-generated reward to learn a useful set of skills. It can cope with situated learning and has successfully demonstrated learning useful behaviors [11], and a merger with ER as in [12] was able to demonstrate navigation in a simulated environment. However, embodiment in a physical world produces a high dimensional continuous space problem that may be beyond the IMRL’s current capability.

There has been a large amount of research in model learning as in [13] and [14], yet the majority focus on action, sensor or internal models of the robot and not the external world. Furthermore, the ability to predict and be “surprised” has not been fully exploited by such learning. However, the use of “surprises” has been used in some applications such as traffic control [15] and computer vision [16].

Finally, learning a world model may be accomplished by simply recording all the interactions with the environment using conventional map building such as Simultaneous Localization and Mapping (SLAM) [17]. However, these approaches require accurate-enough models for the robot’s actions and sensors to start with.

2. Overview of Surprise-Based Learning

The dictionary defines “surprise” as “a sudden or unexpected encounter”. In Surprise-Based Learning (SBL) there is a surprise if the latest prediction is noticeably different from the latest observation. The algorithm must not only detect a surprise, it must also distinguish a possible cause for the surprise by investigating the change in features. Section 4 discusses how a surprise is encountered and handled by SBL.

The algorithm follows the framework visualized in Fig. 1.

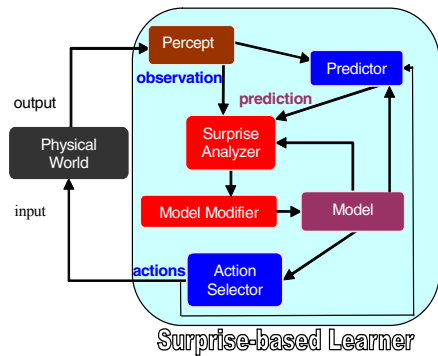


Figure 1. Surprise-based Learning Framework

This framework can be abstractly described as follows: After performing an action, the world is sensed via the perceptor module which extracts feature information from one or more sensors. If the algorithm had made a prediction, the surprise analyzer will validate it. If the prediction was incorrect, the model modifier will adjust the world model accordingly. Based on the updated model the action selector will perform the next action so as to repeat the learning cycle.

The current implementation of the SBL framework is on a SuperBot modular robot [18] with bi-directional WiFi communication, a camera, a short distance range sensor and an external PC capable of wirelessly interfacing with the robot and its sensors. The robot is placed inside a large box which has 4 uniquely colored walls and a discernable floor, as seen in Fig. 2a), which serves as a good structured environment for testing SBL. The forward facing camera is mounted in a way that the robot loses sight of the ground plane when it is approximately 6" from the wall it's facing. The range sensor has also been adjusted so that it's facing the same direction as the camera and its maximum range response is roughly 10".

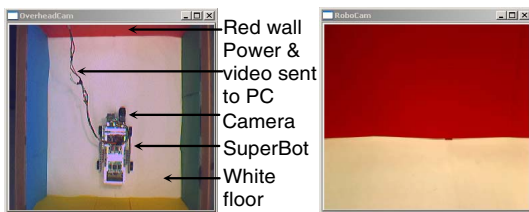


Figure 2. a) Experimental setup b) View from robot

The robot is preloaded with 4 actions corresponding to consistent movement in 4 directions, namely forward, backward, turn left and right. Note that the names for the actions have been chosen to reflect human understanding, yet they are merely labels which are meaningless to SBL and can very well be relabeled as "action1", "action2" etc. The vision and range sensor data are relayed to the PC where the learning algorithm is invoked and a suitable action is radioed back to the robot. The external PC is required due to the limited processing power and memory onboard a single modular robot, but in future SBL will be entirely on the robot, possibly by distributing it amongst several modules. Vision processing

using mean shift segmentation [19] and hue, saturation, value color matching enables the robot to uniquely identify features, label and match them without any a priori knowledge of the environment.

A set of comparison operators are provided for surprise analysis, in particular operators are required to detect the presence (%) or absence (~) of a feature, the change in the size of a feature (<, <=, =, >=, > representing less than, less than or equal, equal, greater than or equal, and greater than respectively), the difference in the distance reading and the displacement of the center of each feature with respect to the frame of reference (horizontal displacement is indicated by the 'x' prefix and vertical displacement by 'y').

The demonstration of SBL will be a sequence of experiments as follows:

- 1) The robot is arbitrarily placed into the environment without any prior knowledge about its position and orientation;
- 2) The robot is given a "goal scene" to seek. The position and orientation of the goal scene is not known, and the scene may be completely unfamiliar to the robot;
- 3) The robot learns a set of prediction rules from the environment by exploration and experiencing surprises;
- 4) When sufficient rules are learned, the robot plans its actions to navigate itself to the goal scene. For example, if the goal scene is "a corner of red and blue walls", then the robot will move itself in the environment so that it will end at the position where its camera will see the goal scene.

As is visible, such demonstrations require the robot to have abilities for (1) problem solving, (2) learning based on surprises, (3) exploration based on current prediction rules, (4) dynamically accepting new goals on the fly, and (5) continuous learning based on surprises if the environment changes.

3. Prediction Model

In SBL, the world model is represented as a set of rules:

$$\text{Rule} \equiv \text{Conditions} \rightarrow \text{Action} \rightarrow \text{Predictions} \quad (1)$$

$$\text{Condition} \equiv (\text{Feature} \rightarrow \text{Operator} \rightarrow \text{Value}) \quad (2)$$

$$\text{Prediction} \equiv (\text{Feature} \rightarrow \text{Operator}) \quad (3)$$

Conditions are comprised of one or more logical statements, each describing the state of a perceived feature in the environment prior to the execution of an action. A condition can be represented as a triple containing a feature identifier, a comparison operator and a comparison value as in (2). In a condition when an operator is applied to a feature, it provides a comparison against a value which qualifies the applicability of the associated rule. i.e. $\text{Condition1} \equiv (\text{feature1}, >, \text{value1})$ means that Condition1 is true if feature1 is greater than value1. In this model several logically related conditions can be grouped together to form a clause using 'And' and 'Not' logical operators. Predictions are logical clauses that describe the expected change in features as a result of performing an action. As seen in (3) a prediction can be represented using a tuple containing a feature and a comparison operator which indicates the change. i.e.

Prediction1 \equiv (feature1, >) means that if the rule is successful the value of feature1 will increase. In order to minimize the number of experiments required to update rules appropriately, all sensor data related to the last (or past) successful instance or invocation of a rule is recorded against the rule. This includes storing the observed state of the environment prior to performing the action hereby referred to as the “before past” (i.e. “the conditions before a past application of the action”) as well as storing the resulting state of environment known as “after past” (i.e. “the observation after a past application of the action”). The use of this additional information will be discussed later.

The basic model used for the experiments presented here does not yet accommodate probability, as it assumes that any major changes in the environment are detectable via the sensors without any probabilistic branching effects caused by ambiguity. This is ensured by adding more sensors and comparison operators to the surprise analysis process, should ambiguity arise. The model also assumes that a set of discrete actions are provided to the learner such that low level motor commands need not be learned, but the meaning or outcome of each action is unknown. In other words the robot does not have a comprehensive action model which states that a particular action would result in displacement by a certain distance in a certain direction. The final assumption made by the model is that comparison operators are provided to the surprise analysis process with a predefined priority so as to distinguish the most significant change.

4. Surprise-Based Learning Algorithm

The SBL algorithm follows a cycle which includes prediction, surprise detection and analysis, rule creation, selection, validation, maintenance via rule splitting & rule refinement, rule abstraction and planning. The details of the algorithm can be found in [20] but we describe the component procedures as follows:

4.1. Surprise Analysis

Surprise analysis is to attempt to identify the most likely reason for a surprise. This is difficult as a prediction could contain one or more features and these features could have complex relations to other features. Our approach works by taking each feature observed in a particular instance and comparing it to all the features observed in another instance using comparison operators. The comparison would yield existential and quantitative differences in the features. Note that SBL does not record every feature change, instead surprise analysis terminates as soon as one difference is encountered, because it prefers to create more general rules that can be applied, surprised, and refined in the future. Therefore, the order or priority in which comparison operators are applied on the features and their attributes determines the generality of a prediction rule. In turn, surprise analysis impacts the quality of the world model, as the total number of rules required to model the physical environment is directly coupled to the algorithm’s ability to identify the exact cause of

a surprise. This will be elaborated in the experimental results later.

4.2. SBL Logic

Rule Creation:

$$\text{Rule 1} = C_1 \rightarrow \text{Action} \rightarrow P_1 \quad (4)$$

Rule Splitting with Prediction Modification:

(C_2 being the reason for the surprise, and P_2 a second prediction which may or may not exist – see subsection 4.5)

$$\text{Rule 1.1} = C_1 \wedge C_2 \rightarrow \text{Action} \rightarrow P_1 \vee \neg P_2 \quad (5)$$

$$\text{Rule 1.2} = C_1 \wedge \neg C_2 \rightarrow \text{Action} \rightarrow \neg P_1 \wedge P_2 \quad (6)$$

After splitting, the 2 rules are marked as “complementary”.

Rule Refinement, given Rule 1.1 failed:

(C_3 being the reason for surprise)

$$\text{Rule 1.3} = C_1 \wedge (C_2 \wedge C_3) \rightarrow \text{Action} \rightarrow P_1 \vee \neg P_2 \quad (7)$$

$$\text{Rule 1.4} = C_1 \wedge \neg (C_2 \wedge C_3) \rightarrow \text{Action} \rightarrow \neg P_1 \wedge P_2 \quad (8)$$

Rule Refinement, given Rule 1.2 failed:

(C_3 being the reason for surprise)

$$\text{Rule 1.3} = C_1 \wedge (\neg C_2 \wedge C_3) \rightarrow \text{Action} \rightarrow P_1 \vee \neg P_2 \quad (9)$$

$$\text{Rule 1.4} = C_1 \wedge \neg (\neg C_2 \wedge C_3) \rightarrow \text{Action} \rightarrow \neg P_1 \wedge P_2 \quad (10)$$

4.3. Rule Creation

A rule is created as in (4) by performing an action and recording a significant change in the environment as the prediction (P_1) and its prerequisites as the condition (C_1). Sensor data is processed to extract feature information. Surprise analysis is performed by applying predefined comparison operators to compare feature information available prior to taking an action hereby referred to as the “before current” and feature information present after its execution known as “after current”. If no changes exist, then a new rule cannot be generated. However, if more than one change exists, then only one change must be selected and recorded in the rule using surprise analysis. The current instance is recorded against the rule as the last successful instance for future reference.

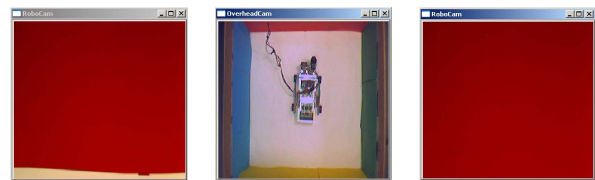


Figure 3.a) Floor b) Forward place c) ~ WhiteFloor

For example consider executing the forward action when the robot is located as in Fig. 2a). Rule creation or CreateNewRule() compares “before current” sensor data seen in Fig. 2b) and the “after current” sensor data seen in Fig. 3a) resulting in Rule 1. The backward action from Fig. 3a) to Fig. 2b) results in Rule 2.

Rule 1: (RedWall, %, 0) \rightarrow FORWARD \rightarrow (RedWall, >)

Rule 2: (RedWall, %, 0) \rightarrow BACKWARD \rightarrow (RedWall, <)

4.4. Rule Selection & Rule Validation

When the robot decides on an action either randomly or with the aid of the planner, the rule selection process filters the stored prediction rules by this action. It then matches the current state of the environment against the conditions of each of these rules so as to return only the applicable prediction rules. For example, rule selection returns Rule 1, if the forward action is selected in Fig. 3a). Should no matching rule exist, a new rule will be generated after the execution of the action. If a match is found, the action will be performed and the validity of predictions of the selected rule will be ascertained by considering the new state of the environment. This is known as rule validation. Rule validation returns true if all predictions are satisfied, meaning there is “no surprise”, otherwise there is a “surprise” and rule maintenance is invoked.

4.5. Rule Splitting

Rule splitting is performed when a surprise occurs and the surprise is caused by a rule that has never been split before. Rule splitting works by comparing the last successful instance of the rule against the currently unsuccessful instance. A new set of conditions (C_2) that were present prior to the failure can be extracted by comparing “before past” and “before current.” In SBL when splitting it is important to generate a prediction for each feature stated in the conditions. Hence, if the new conditions refer to a new set of features, then a new set of predictions (P_2) will be generated by comparing “after current” and “before current”, while preserving the original set if predictions (P_1). However, if the new conditions refer to features already included in the original set of predictions then a new set of predictions will not be created. The original failed rule is altered as in (5) and a new complementary rule is created as in (6) using negation on the newly identified set of conditions ($\neg C_2$), while ensuring that the predictions remain complementary, i.e. $\neg (P_1 \vee \neg P_2) = (\neg P_1 \wedge P_2)$. The complementary rules are flagged such that if either of them fails in future they are both altered together by rule maintenance using the rule refinement process. After splitting the last successful instance of the newly created rule is updated, while leaving the original rules’ history unaffected.

To better understand the generation of the new set of predictions (P_2), consider the following scenarios. Executing forward from the location in Fig. 3b) where the view is Fig. 3a) results in Fig. 3c), where Rule 1 is valid. A subsequent forward action produces a surprise as the floor has disappeared and the wall remains constant. This splits the original Rule 1 and creates the complement Rule 3 as follows:

Rule 1: (RedWall, %, 0) AND (WhiteFloor, %, 0) \rightarrow FORWARD \rightarrow (RedWall, >) OR NOT (WhiteFloor, ~)

Rule 3: (RedWall, %, 0) AND NOT (WhiteFloor, %, 0) \rightarrow FORWARD \rightarrow (RedWall, <=) AND (WhiteFloor, ~)

In contrast, executing backward from Fig. 2a) several times results in the floor and wall remaining constant after the robot

hits the rear wall. This splits the original Rule 2 and creates the complement Rule 4 without a new prediction as follows:

Rule 2: (RedWall, %, 0) AND (RedWall, >, Value2) \rightarrow BACKWARD \rightarrow (RedWall, <)

Rule 4: (RedWall, %, 0) AND NOT (RedWall, >, Value2) \rightarrow BACKWARD \rightarrow (RedWall, >=)

4.6. Rule Refinement

As learning continues if one rule belonging to a complementary pair such as (5) failed, rule refinement performs an update to the original rule by extracting a new conditions (C_3) through same comparison process as described in rule splitting. Yet, new predictions are not generated as the predictions will no longer be altered. The addition of new conditions to the original rule results in specialization as in (7). The complementary rule is recreated as in (8) by negating the second clause $\neg (C_2 \wedge C_3)$ of the updated failed rule, affectively generalizing the rule to maintain its complementary relationship. To elaborate this further, the results of refining the complement of a split rule as in (6) are shown in (9) and (10). Once again only the last successful instance of the complementary rule is updated. Subsequent refinements will be subjected to the same process outlined above.

For example consider executing the backward action from Fig. 3c), which results in Fig. 3c) again, instead of Fig. 3a) when the robot is close to the red wall and given that Rule 2 has been split. Comparing “before past” and “before current”, indicates that the floor remains missing and there is no change in the wall. Therefore, Rule 2 and its complement Rule 4 will be refined as follows:

Rule 2: (RedWall, %, 0) AND (RedWall, >, Value2) AND (WhiteFloor, %, 0) \rightarrow BACKWARD \rightarrow (RedWall, <)

Rule 4: (RedWall, %, 0) AND NOT ((RedWall, >, Value2) AND (WhiteFloor, %, 0)) \rightarrow BACKWARD \rightarrow (RedWall, >=)

4.7. Planning & Rule Abstraction

SBL is not complete without a planner. The planner’s implementation is independent of rest of the algorithm, meaning that it could be designed using a greedy search. In essence the planner must simply return a sequence of rules that are to be executed to reach a goal scene comprised of features, from the current state. This can be achieved as the predictions of one rule could satisfy the conditions of the next rule, forming the desired sequence of actions.

One difficulty of this planning is that the information relating the transition from one feature to the other could be buried complexly within the pair of complementary rules. To overcome this problem, information abstraction or inference is applied to complementary rules so that we can extract the required feature transition information into a new abstract rule for planning purposes. For example consider the complementary Rule 1 and Rule 3. By taking the logical intersection of the conditions and the logical difference of the predictions, it is possible to infer Rule 5 that states a

relationship between two features, and in particular identifies the action required to transition from one feature to the other. Hence, rule abstraction is a powerful tool to infer feature relations such as corner transitions.

Rule 5: (RedWall, %, 0) → FORWARD → (WhiteFloor, ~)

To minimize the execution time of the planner, rule abstraction is performed immediately after rule splitting. Abstract rules are marked such that rule selection will not use them as predictions, as they are only required for planning.

The robot can be given a target or goal scene either prior to, or during, or after the learning process. Runtime goal assignment permits SBL to change goals dynamically. In fact, a plan can be considered a set of sub goals that leads to the goal scene. When planning is invoked, the next action from the plan subsumes any random action that the robot had previously decided to explore.

5. Experiment Results & Analysis

Table 1. Components of each set of experiments

Set	Actions	Sensors	Operators	Selection
A	6	Vision, Range	%, ~, x>, x<, >, <, =	Biased
B	4	Vision, Range	%, ~, x>, x<, >, <, =	Biased
C	4	Vision, Range	%, ~, >, <, =, x>, x<	Biased
D	4	Vision, Range	%, ~, >, <, =	Biased
E	4	Vision, Range	%, ~, >, <, =	Human
F	4	Vision, Range	%, ~, >, <, =	Random
G	4	Vision	%, ~, >, <, =	Random
H	4	Vision	%, ~	Random

Table 2. Results for each set of SBL experiments

Set/ No	Actions			Surprises			Rules			Preds	Goal
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Avg	%
A/6	46	85	110	4	37	50	22	39	44	215	84
B/6	70	92	120	16	42	72	15	28	47	248	50
C/2	62	74	85	37	48	59	28	35	41	115	0
D/6	54	88	134	28	51	62	23	44	56	168	33
E/4	46	64	102	14	27	31	14	32	42	146	50
F/6	64	90	150	44	67	89	28	42	52	212	17
G/3	80	90	100	52	60	72	16	20	24	172	0
H/1	20	20	20	16	16	16	4	4	4	4	0

Eight sets of experiments were conducted to establish the feasibility of SBL in solving this problem, to demonstrate that the environment was sufficiently complex to rigorously test most aspects of SBL, and to determine the parameters that affect the performance of SBL. Each subsequent set of experiments was conducted by adjusting some of the constraints such as the number of actions, the order of surprise analysis, the number of comparison operators, the amount of bias from human influence and the number of sensors.

Table 1 displays the name of the set, the number of actions available to the robot, the list of sensors on the robot, a set of comparison operators used for feature comparison listed according to their priority or order, and the action selection criteria. The action selection criteria, indicates whether any randomly generated actions were biased, completely random or overridden by a human operator. Note that the listed comparison operators are applied during rule creation, yet during rule splitting and refinement the remaining

complementary operators are used to maintain complementary rules. Table 2 displays the results for each set of experiments highlighted in Table 1. The total number of experiments, actions, surprises, rules and the percentage of experiments that accomplished the goal are marked here. Within a set, multiple experiments were conducted by varying the starting location and orientation of the robot, along with performing runtime goal assignment to a random scene, prior to commencing learning, after a short period of learning or after the entire environment was traversed. Videos for some of the experiments are available at www.isi.edu/robots/media-surprise.html.

Set-A represents the most successful experiments. The robot was able to learn accurate prediction rules over a period of time, resulting in surprises receding and prediction successes continuously increasing. Additional experiments were carried out by toggling the actions (such as switching forward with backward) and inverting the sensors (such as a horizontal flip of the image, which is similar to switching left and right). Given a sufficient amount of time SBL was able to successfully learn this environment even under these conditions. About 84% goal rate indicates that SBL did not reach the goal once during experiment set-A. The reason for this was because the planner was invoked before the learner was able to capture all the necessary relations. However, all experiments reach the goal after the rules are learned.

The rest of Table 2 shows the effects of SBL components on the learning performance. In Set-B, we change the number of actions. In Set-C, we vary the priority of operators in surprise analysis. In Set-D, we change the number of operators. In Set-E, we bias the selection of actions to guide more fruitful exploration. In Set-F, we remove the bias to see the effect on the rate of learning. Set-G changes the number of sensors, and Set-H reduces the number of sensors and operators until the information becomes insufficient for proper learning. Details of these experiments can be found in our technical report [20].

The experiments also show how SBL deals with fluctuations in the image recognition under different lighting conditions. Since, features are identified, labeled and matched dynamically there are occasions where bad lighting such as shadows cause a feature to be classified incorrectly. Regardless of misclassification, as long as the classification remained consistent throughout the course of the experiment, SBL would learn to overcome the surprises by generating accurate prediction rules. Note also that due to the complementary nature of rule splitting, a specialized rule can have very precise predictions, but its complementary rule will be general enough for further learning. Such rules are marked during rule selection. In addition, all new rules are compared against the stored rules so as to avoid duplications.

From the results, it is evident that SBL can learn the model of the environment, and the quality of the model varies with the number of actions, surprise priority, number of comparison operators, biased actions, and number of sensors.

6. Conclusion & Future Work

This paper presents surprise-based learning that enables a robot to reach a goal by navigating an unknown environment based on its actions and sensors but doing so without a priori knowledge about the consequences of actions. Thus, the learner can adapt to unexpected and dynamic changes in its actions and sensors, and can deal with new environments. At present, SBL has successfully demonstrated that a mobile robot placed in a static environment can learn through exploration to navigate to a particular scene, which can be assigned or changed during runtime. The algorithm does not discretize the environment and is grounded in a physical environment where elements such as the robot's position cannot be reset. SBL does not yield an "exact" map of the environment; instead the model is abstract but accurate enough for a robot to accomplishing its task. Learning and planning occurs in a feasible amount of time making SBL suitable for a mobile robot.

In the future, SBL will be tested on a robot placed in a large static environment such as an office room. Probabilistic branching will be added to prediction rules or the selection process, so as to accommodate dynamic environments and the ability to deal with ambiguity given that sufficient information is unavailable via the sensors. Rules will also be assigned a success rate such that rule rejection or "forgetting" could be incorporated to facilitate relearning. Rule abstraction can be improved to recognize similarities in rules that would mark the similarities in features, which in turn replaces the common rules with an abstract representation (i.e. recognizing that there are four identical features such as the four colored walls and creating abstract rules that replace these common rules). The current implementation focuses on learning features, but this must be expanded to identify objects (ball, box etc.) and possibly concepts (corner, wall etc.) in order to enrich its interactions in real world situations. Finally, the possibility of learning the surprise analysis priority and application of SBL to other domains will also be investigated.

7. ACKNOWLEDGMENT

This research is funded by AFOSR grant FA9550-06-0336, but the opinion in this paper is solely by the authors. We are grateful to Rizwan Khan and Feili Hou for conducting some experiments and all the members of USC/ISI Polymorphic Robotics Laboratory for their feedback and support.

8. REFERENCES

- [1] Piaget J., "The Origins of Intelligence in the Child", Norton, 1952.
- [2] Simon H., Lea G., "Problem solving and rule induction: A unified view", Knowledge and Cognition, Hillsdale, NJ, 1974.
- [3] Cohen P., Atkin M., Oates T., Beal C., "Neo: Learning conceptual knowledge by sensorimotor interaction with an environment", Intl. Conference on Intelligent Agents, 1997.
- [4] Hurang X., Weng J., "Novelty and reinforcement learning in the value system of developmental robots" 2nd Intl. Workshop on Epigenetic Robotics, 2002.

- [5] Kaelbling L., Littman M., Moore A., "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research, 1996, Vol. 4.
- [6] Nolfi S., Floreano D., "Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines", MIT Press, Cambridge, MA, 2000.
- [7] Stout A., Konidaris G., Barto G., "Intrinsically Motivated Reinforcement Learning: A Promising Framework For Developmental Robot Learning", AAAI Spring Symposium on Developmental Robotics, 2005.
- [8] Shen W.-M., "Autonomous Learning From The Environment", New York, NY: W.H. Freeman and Company, 1994.
- [9] Shen W.-M., "Learning from the Environment Based on Actions and Percepts" Ph.D. dissertation, Dept. Computer Science., Carnegie Mellon University., Pittsburgh, PA, 1989.
- [10] Lipson H., Bongard J., "An Exploration-Estimation Algorithm for Synthesis and Analysis of Engineering Systems Using Minimal Physical Testing", ASME Design Engineering Technical Conferences, Salt Lake City, UT, 2004.
- [11] Oudeyer P., Kaplan F., Hafner V., "Intrinsic Motivation Systems for Autonomous Mental Development", IEEE Transactions on Evolutionary Computation, 2007, Vol. 11.
- [12] Schembri M., Mirolli M., Baldassarre G., "Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot", IEEE International Conference on Development and Learning, 2007.
- [13] Pierce D., Kuipers B., "Map learning with uninterpreted sensors and effectors", Artificial Intelligence, 1997, 92: 169-229.
- [14] Stronger D., Stone P., "Towards Autonomous Sensor and Actuator Model Induction on a Mobile Robot", Connection Science, 18(2), June 2006, pp. 97-119.
- [15] Horvitz E., Johnson A., Sarin R., Liao L., "Prediction, Expectation, and Surprise: Methods, Designs, and Study of a Deployed Traffic Forecasting Service", Twenty-First Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 2005.
- [16] Itti L., Baldi P., "A Surprising Theory of Attention", IEEE Workshop on Applied Imagery and Pattern Recognition, October 2004.
- [17] Thrun, S., "Robotic Mapping: A Survey", Exploring Artificial Intelligence in the New Millennium, Morgan Kaufmann, 2002.
- [18] Salemi B., Moll M., Shen W., "SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System", IEEE Intl. Conf. on Intelligent Robots and Systems, Beijing, October 2006.
- [19] Comaniciu D., Meer P., "Mean Shift: A Robust Approach toward Feature Space Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, May 2002, Vol. 24, pp. 603-619.
- [20] Ranasinghe N., Shen W.-M., "The Surprise-Based Learning Algorithm", USC ISI internal publication, April 2008.