Surprise-Based Developmental Learning and Experimental Results on Robots

Nadeesha Ranasinghe and Wei-Min Shen, *Member, IEEE*

***Abstract*—Learning from surprises and unexpected situations is a capability that is critical for developmental learning. This paper describes a promising approach in which a learner robot engages in a cyclic learning process consisting of "prediction, action, observation, analysis (of surprise) and adaptation". In particular, the robot always predicts the consequences of its actions, detects surprises whenever there is a significant discrepancy between the prediction and the observed reality, analyzes the surprises for causes, and uses the analyzed knowledge to adapt to the unexpected situations. We tested this approach on a modular robot learning how to navigate and recover from unexpected changes in sensors, actions, goals, and environments. The results are very encouraging.**

***Index Terms*—Learning systems, unsupervised learning, surprise-based learning, developmental robotics.**

I. Introduction

A challenging issue for autonomous learning systems is to deal with unexpected situations. These situations can range from surprising changes in either robots themselves (such as sensors, actions, and body configurations), or the tasks and environments they are facing. These surprises are unavoidable because no matter how carefully a system is designed and engineered at the outset, its initial knowledge is bound to be incomplete or incorrect with respect to the richness (continuous, uncertain, and vast information space) of the real world. Hence, an autonomous system must learn and adapt. It must detect surprises, analyze surprises, and adapt its knowledge whenever possible. This requirement demands an integrated solution for perception, memory, learning, reasoning, focusing of attention and discovery. In [1] we introduced Surprise-Based Learning (SBL) as an integrated architecture for such learning. This paper reports some significant new findings and experimental results of SBL.

Three advantages of the SBL approach are experimented and observed: (1) scalability to the number of sensors and actions, (2) fault-tolerance for failures and malfunctions in sensors & actions, and (3) adaptability to unexpected changes and uncertainties in tasks and environments. These advantages are enabled by three new underlying technologies: an extended algorithm for complementary discrimination learning, a focus-of-attention technique for automatically determining relevance and associating sensors with actions, and a memory/forgetting mechanism for model management and goal transfer.

Scalability is measured by the size of learned model with increasing number of sensors and actions. To test the robot's ability to react to unexpected faults, we deliberately twisted the robot's camera and exchanged the consequences of its actions. Transfer learning is tested by allowing the robot to learn how to reach a hidden goal via exploration and then secretly relocate the goal to various different locations. All these changes are unexpected by the robot beforehand. We observe that the robot is first surprised by these changes and then gradually "forget" the obsolete knowledge and learn the necessary new knowledge and adapt to the new situations. This quick adaption is also observed as a form of knowledge transfer from the old goals to the new goals.

The paper is organized as follows. Section 2 outlines the background for surprise-based learning. Section 3 provides a description of the experimental setup. Section 4 describes the integrated architecture, the SBL process and the underlying techniques mentioned above. Section 5 presents the experimental results. Section 6 concludes the paper with a discussion and future research directions.

II. Background and Related Work

The basic concept of surprise-based learning was first proposed by Shen & Simon and later formalized as Complementary Discrimination Learning [2]. Over the years, researchers have attempted to formalize this intuitively simple but powerful idea into an effective and general learning technique. A number of experiments in discrete or symbolic environments have been carried out with success [3]. This paper generalizes these previous results for real robots to learn autonomously from continuous and uncertain environments.

In computer science, SBL is related to several solutions for the inverse problem, such as Gold's algorithm, L* algorithm, L*-extended and D* algorithm. For learning from stochastic environments, SBL is related to learning hidden Markov models, partially observable Markov decision processes, predictive state representations and temporal difference algorithms. Some systems also incorporate *novelty* [4], with a flavor of surprise, into the value function of states, although they are not used to modify the learned models. The notion of *prediction* is common in both developmental psychology and AI. Piaget's *Constructivism* and Gibson's *affordance* are two

Manuscript received Feb. 8, 2009. This work was supported by AFOSR grant FA9550-06-0336, but the opinion in this paper is solely by the authors.

N. Ranasinghe (e-mail: nadeesha@isi.edu) and W.-M. Shen (e-mail: shen@isi.edu) are with the Information Sciences Institute, University of Southern California, Marina Del Rey, CA 90292 USA.

978-1-4244-4118-1/09/$25.00 ©2009 IEEE

famous examples. In AI systems, the concepts of *schemas* and *fluent* all resemble the SBL *prediction rules*, although their primary use is not for detecting and analyzing surprises. SBL has previously been compared to many different learning techniques in [5] such as reinforcement learning (RL) and evolutionary robotics [6, 7]. The results presented in this paper confirm some advantages of model-based learning for quicker adaptation, better scalability, and dealing with continuous space/time problems with uncertainties.

III. Experimental Setup

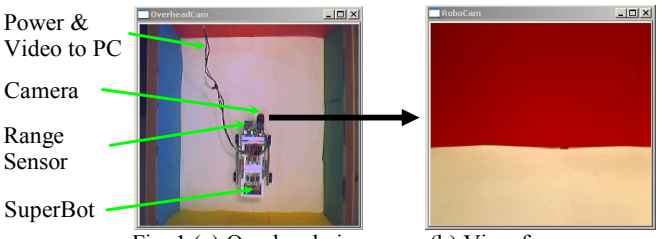

Fig. 1 (a) Overhead view (b) View from camera

SBL is implemented and tested on a single SuperBot module [8]. The robot is equipped with WiFi communications, an onboard camera and a short distance range sensor. An external PC with an overhead camera is used to perform heavy-duty computational tasks. The robot lives in an environment with four uniquely colored walls and a discernable floor as seen in Figure 1.

The robot is preloaded with four actions corresponding to the motions of forward, backward, left and right, but it has no priori knowledge about the expected results of these actions. Each action is executed for 5 seconds and terminated. The onboard camera and range sensor allow the robot to make an *observation* (or *scene*) before and after each action. An observation consists of a set of identifiable *percepts* (or *sensor features*) that include blobs of colors, the size and center-location of visible blobs, and the readings of the range sensor.

Both sensors and actions are uncertain and noisy due to the intrinsic nature of the real world (even though this is a small environment). For example, sensors are nondeterministic and may misclassify or return incorrect values, and actions may produce unexpected results due to slippage.

The robot has no pre-knowledge about the percepts or their relations to other percepts and actions. During the scene analysis, the percepts are reasoned and compared by a set of mental operators that are given to the robot at the outset. In contrast to previous publications [1, 5] an important difference of SBL presented here is that there is no assumed order of these operators. The current operators include the presence (%) or absence (~) of a feature, the change in the size of a feature (↓, ↓=, =, ↑=, ↑), the difference in the distance reading, and the center-location or displacement of each visual feature with respect to the frame of reference (x↑, x↓ refer to the x-coordinate of the feature increasing and decreasing respectively, while y↑, y↓ refer to the y-coordinate increasing and decreasing respectively).

IV. Surprise-Based Learning

A. Integrated Architecture

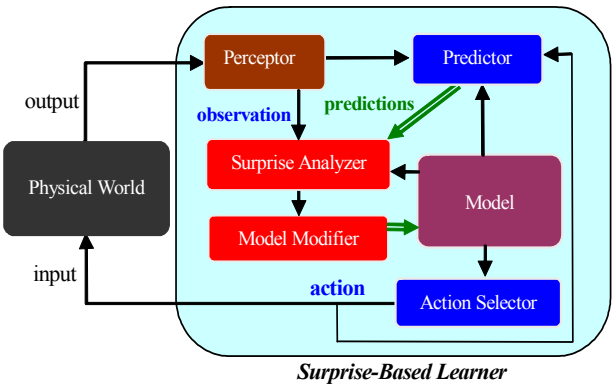

Fig 2. SBL Architecture

The SBL architecture and its learning cycle are shown in Figure 2. The knowledge of the robot is represented in the "model" as a set of probabilistic prediction rules. Actions are planned using a generalized mean-end planner ("action selector") or randomly selected if the current model offers no relevant prediction rules. The "predictor" returns all prediction rules whose conditions and actions match the current state of the environment and the selected action. The action is executed and the "preceptor" will perceive the resulting state as the new current state. If no prediction was made, a new rule is created to reflect the observed conditions and consequences of the action. If all predictions are satisfied by the new observations, then the cycle continues. Otherwise, a surprise is detected and the "surprise-analyzer" and "model-modifier" will revise the model to reflect the knowledge learned from the surprise.

In this paper, a single failure can result in a surprise, but in general the detection of surprise is based on probability P(Prediction|Condition,Action). The prediction rules are organized in complementary pairs to maintain the flexibility to be generalized or specialized/discriminated. Probabilities of the rules are implicitly implemented in the rule revision process. The learned knowledge is represented in the model consisting of prediction rules in the format of (1) below.

$$\text{Rule} \equiv \text{Conditions} \rightarrow \text{Action} \rightarrow \text{Predictions} \quad (1)$$
$$\text{Condition} \equiv (\text{Feature} \rightarrow \text{Operator} \rightarrow \text{Value}) \quad (2)$$
$$\text{Prediction} \equiv (\text{Feature} \rightarrow \text{Operator}) \quad (3)$$

Each prediction rule is a triplet as in (1) of "condition", "action", and "prediction". Conditions are logical statements that describe the state of the perceived features prior to the execution of an action, but differ from traditional RL in that they are learned and can represent a set of states. A condition can be represented as a triple as in (2). Using the comparison operators given to the robot, each condition describes a relationship of a feature and its value. For example, the expression Condition1 ≡ (feature1, ↑, value1) means that Condition1 is true if feature1 is greater than value1. Several logically related conditions can be grouped together to form a clause using 'And' and 'Not' logical operators. Predictions are

clauses that describe the expected change in the state of the perceived features as a result of performing an action. As seen in (3) a prediction can be represented using a tuple. e.g. Prediction1 ≡ (feature1, ↑) means that if the rule is successful the value of feature1 will increase.

A surprise occurs if something unexpected happens. However, a surprise can be either "good", if the unexpected result is desirable, or "bad" otherwise. An important aspect of prediction rules is that they can be sequenced to form a prediction sequence. This can be used to represent many other concepts such as "plan," "exploration," "experiment," "example," and "advice" in a unified fashion [3].

B. Complementary Discrimination Learning

During learning and adaptation, prediction rules are created and refined according to a set of templates described in equations (4)-(10) below.

Creation of a Single Rule: *Let C_0 and P_0 represent the change of a percept before and after a newly explored action, a new rule is created as follows:*

$$\text{Rule}_0 = C_0 \rightarrow \text{Action} \rightarrow P_0 \quad (4)$$

Creation of Complementary Rules: *If a surprise is caused by a single rule (e.g. $Rule_0$ above) for each possible cause $\neg C_X$ identified by the analysis of the surprise, a pair of complementary rules is created, where P_X is a newly observed consequence of the action with respect to $\neg C_X$ as follows:*

$$\text{RuleA}_1 = C_0 \wedge C_X \rightarrow \text{Action} \rightarrow P_0 \vee \neg P_X \quad (5)$$

$$\text{RuleB}_1 = C_0 \wedge \neg C_X \rightarrow \text{Action} \rightarrow \neg P_0 \wedge P_X \quad (6)$$

Notice that the two rules are complementary to each other in condition & prediction. $RuleA_1$ reflects the previous success and $RuleB_1$ reflects the new surprised situation.

Refinement of Complementary Rules: *If a surprise is caused by a RuleA that has a sibling RuleB, where φ_C represents the rule's current condition minus C_0, and η_P represents the prediction of the rule, as follows:*

$$\text{RuleA} = C_O \wedge \varphi_C \rightarrow \text{Action} \rightarrow \eta_P \quad (7)$$

$$\text{RuleB} = C_O \wedge \neg \varphi_C \rightarrow \text{Action} \rightarrow \neg \eta_P \quad (8)$$

$RuleA_1$ & $RuleB_1$ are instances of this format. Then for each possible cause $\neg C_Y$ identified by the analysis of the surprise, the rules will be refined as follows:

$$\text{RuleA} = C_O \wedge \varphi_C \wedge C_Y \rightarrow \text{Action} \rightarrow \eta_P \quad (9)$$

$$\text{RuleB} = C_O \wedge \neg (\varphi_C \wedge C_Y) \rightarrow \text{Action} \rightarrow \neg \eta_P \quad (10)$$

Notice that (7)-(10) can be applied to any pair of complementary rules. In general, a pair of complementary rules can be refined multiple times so that as many C_Y can be inserted into their conditions according to this procedure. Whenever a rule is discriminated, its complementary rule will be generalized, hence the name complementary discrimination learning.

Please refer [1] for more details on sections 4.2.1, 4.2.2, 4.2.3, 4.2.4, 4.3 as they are only summarized in this paper for completeness. In contrast, this paper generalizes these concepts as it does not assume any ordering of comparison operators in the analysis of surprise.

I. Creation of a Single Rule

If the robot is exploring an action that has never been executed before, then there is a chance that new rules will be created. SBL compares the percepts in the scenes before and after the action using the given comparison operators. If no changes can be found by these comparisons, then no further action is taken. Otherwise, for each change detected, a new rule is created according to template (4), with P_O being the changed percept and C_O being the prerequisite.

Fig. 3 (a) Initial view (b) 1st "forward" action (c) 2nd "forward"

As an example of single rule creation, consider the situation in Figure 3 (a) & (b), where the robot first explores the action "forward" and the scenes before and after the action are as shown. There are three percepts in these scenes, red (wall), white (floor), and range. The single rule creation mechanism detects four changes in these percepts and creates four new rules F1, F2, F3, & F4, listed below. For each newly created rule, the robot records the two scenes before and after the action and refers to them in the future as the rule's "base-condition-scene" and "base-consequence-scene".

RuleF1: (Red, %, 0) → FORWARD → (Red, ↑)
RuleF2: (Red, %, 0) → FORWARD → (Red, y↑)
RuleF3: (White, %, 0) → FORWARD → (White, ↓)
RuleF4: (White, %, 0) → FORWARD → (White, y↑)

II. Surprise Analysis

As prediction rules are incrementally learned and refined, the robot uses them to make predications whenever it can. A surprise is detected if a prediction fails to be realized in the scene after the action. For each surprise, SBL refers to the scenes before and after the action known as the "surprised-condition-scene" and "surprised-consequence-scene".

A surprise is analyzed by applying the comparison operators to comparing the percepts in the base-condition-scene with those in the surprised-condition-scene. The comparison prefers "novel" differences rather than "ordinary", so it first considers the percepts that are not mentioned in the existing rule before it considers those that do. If no novel difference can be found, the analysis will return causes that are considered as "ordinary" (i.e. those that are already mentioned in the current condition of the surprised rule). If not even an ordinary difference can be detected by these comparisons, then the system must discover hidden features [3] although this topic is outside the scope of this paper. After differences are detected, the analysis will return a set of C_X as the possible "causes" for the current surprise. Each cause is an expression of a percept that was true in the base-condition-scene but false in the surprised-condition-scene.

III. Creation of Complementary Rules

Complementary rule creation is performed when a surprise occurs and is caused by a single rule which does not belong to

a complementary pair. For each identified possible cause C_X of the surprise, a new pair of complementary prediction rules will be created to reflect both the original consequence and the newly observed and surprised consequence P_X. C_X will be appended to the existing condition of the original rule so that the rule is specialized. At the same time, the system will create a new complementary rule by adding the negation of C_X in the condition so that it is generalized. For completeness, these two new rules are ensured to have complementary predictions.

To illustrate complementary rule creation, consider executing the forward action in Figure 3 (c), which caused a surprise in RuleF1 as Red did not increase. The analysis of surprise compares the base-condition-scene in Figure 3 (a) and the surprised-condition-scene in Figure 3 (c), and returns the possible cause as the absence (not present) of White. The complementary rules for RuleF1 are created as follows:

RuleF1.1a:(Red,%,0)∧(White,%,0)→FORWARD→(Red,↑)∨(White,%)
RuleF1.2b:(Red,%,0)∧¬ (White,%,0)→FORWARD→¬(Red,↑)∧(White,~)

IV. Rule Refinement

As learning continues if one of the complementary rules encounters a surprise, then for each now possible cause C_Y identified by the analysis of surprise, the robot will refine the surprised rule and its sibling rule according to equations (7)-(10). C_Y is inserted into the surprised rule for further specialization, while the complementary rule is generalized by negating the entire condition (excluding C_0). The predictions of the rules are not altered.

C. Planning with Abstract Rules

The job of a planner is to find a sequence of rules/actions that when executed can transport the robot from the current state to a goal state. The planner used here is based on standard goal-regression with a greedy search. A goal state is defined as a scene with a set of desired features. To plan effectively the planner extracts the feature transition from a pair of complementary rules and stores the relation as an *abstracted rule*. So from equations (5)-(6) an abstracted rule is defined as:

An Abstracted Rule: $C_0 \rightarrow \text{Action} \rightarrow P_X$ (11)

D. Rule Forgetting

Adaptation requires both learning new and useful knowledge and forgetting obsolete and incorrect knowledge. A prediction rule may be obsolete because the robot's sensors or actions are changed or damaged, or the environment poses new tasks or situations. To deal with unexpected situations, a learning robot should be prepared to consider all possible hypotheses of the relationships of its sensors and actions during its adaptation. Naturally, not all hypotheses are correct and those that are wrong should gradually be forgotten.

Complementary prediction rules are determined to be inappropriate in two ways: they cause contradictions, or consecutively cause surprises. A contradiction occurring immediately after complementary rule creation means that neither of the complementary rules can describe the surprised consequence correctly. This can be caused by two reasons: either the rule was created with a wrong C_0 in the first place (i.e. the feature C_0 was irrelevant to the current action), or the analysis of the surprise returned an incorrect cause C_X, along with an incorrect prediction P_X. Either way, these two rules are self-contradicting and have no future in learning.

The second way to reject rules is to keep track of the progress of a pair of complementary rules. Each time a surprise occurs and complementary rules are refined, it should imply that on subsequent selection of either rule, the robot should produce better predictions. However, if these rules fail consecutively (i.e. 3 failures in a row), then it can be inferred that the feature relation captured in the rule is inappropriate for the given context and that the rules should be rejected.

Fig. 4 (a) Initial view (b) 1st "right" action (c) 2nd "right"

To illustrate rule rejection, consider the scenario where the robot is turning right. Complementary rule creation generates RuleF5.1 & RuleF5.2 as shown below when transitioning from Figure 4 (a) to (b) because the vertical displacement of White remained the same instead of decreasing. On the next right-turn action RuleF5.2 fails as the vertical displacement decreased again, resulting in contradictive failures and the rejection of these complementary rules.

RuleF5.1: (White,%,0)∧(Red,↓,Value5)→RIGHT→(White,y↓)∨¬(Red,↑)
RuleF5.2: (White,%,0)∧¬(Red,↓,Value5)→RIGHT→¬(White,y↓)∧(Red,↑)

From this simple sequence of scenes, it is clear that making any predictions about the White floor while turning is inappropriate as the floor borders all four uniquely colored walls. Thus, rule forgetting facilitates forgetting such inappropriate rules and retaining only useful rules over a period of time.

E. Feature Relevance

Feature relevance is critical for a number of capabilities of an autonomous robot. First, it is critical for learning new knowledge, such as coupling the relevant features with the relevant actions. Second, it is critical for adaptation to unexpected changes in the environment, such as certain features that were previously relevant to certain actions may become irrelevant at some point in time and vice versa. Third, feature relevance is critical for recovering from the damaged or failed sensors and actions. Fourth, feature relevance is critical for the scalability of a learning algorithm, such that the robot can ignore the irrelevant and focus on the relevant to prevent the learning from quickly being saturated.

In SBL, the relevance of features are recorded in a table in which each row uniquely identifies a feature representing the modality of a sensor, an action and an indicator corresponding to its relevance during learning. The relevance flag is updated depending on the following four scenarios:

(1) If a feature does not change its value for any action, then even if the feature remains active it will not be used by SBL.

E.g. a constant valued sensor such as a failed one is ignored.
(2) For a given action, if all rules that contain this feature as its first condition C_0 have been rejected, then the feature is irrelevant to that action. E.g. a random sensor will cause all its rules to fail eventually.
(3) For a given action, if many rules containing the same feature have contradictive failures, then the feature is irrelevant. E.g. the size of the floor is irrelevant for turning.
(4) When none of the active features for a given action register any change (could be hardware change), then consider reactivating these features as potentially relevant again. This is known as "forced relevance", which represents enlarging the attention of the robot. E.g. when the camera is rotated, the confusion will flag many features as irrelevant, but the lack of progress will eventually re-enable them to learn new rules.

SBL performs a lookup on the table and ignores any features that are considered irrelevant during the analysis of surprises for rule creation and refinement.

F. Goal Association and Transfer

During learning, if the robot is shown a "goal scene", it will attempt to generate a plan to reach it. When a goal is not directly observable, it may be sensed by some mysterious sensor when the robot is at the goal location. E.g. suppose the goal is a submerged platform in a body of water and the robot must swim to look for it, then it will know the goal location only when it is standing on the platform [9]. To find such "hidden" goals, the best strategy seems to explore and learn the environment as much as possible while searching. Once the goal is reached, the robot should remember the goal location by associating it with as many visual clues as possible so that it knows what to look for when it revisits the goal. This can be done if the robot has learned a model of the environment.

The strategy used here is to record the associations as a set of "goal scenes" and maintain them in a dynamic list. A new goal scene is added to the list only if a similar scene is not currently in the list. In addition to recording the scenes, two statistics corresponding to the number of successes and failures for each scene are noted. At any time after the initial discovery, if the robot is prompted to go to the hidden goal, the planner is invoked by selecting the best goal scene from the list. The number of successes is incremented each time the robot receives feedback when it encounters a goal scene. Yet, if a goal scene is reached, but no indication of success is received, the number of failures for that goal scene is incremented. The ratio of successes to failures for a goal scene indicates the probability of finding the hidden platform using it. Intuitively, the planner selects the best goal scene by picking the one with the highest probability. If the planner tracks every goal scene in the list and does not come across the hidden goal, SBL concludes that the hidden goal has moved and proceeds to execute random actions. This facilitates learning to reach a hidden goal, as well as transferring knowledge if the location of the goal is secretly moved.

V. Experimental Results

A. Scalability

SBL performance was measured by the time taken to accomplish a particular task (number of actions executed) and the amount of resources consumed (number of active prediction rules). Three scalability experiments were conducted by giving many relevant and irrelevant sensors & actions and observing the size of the learned model in terms of the number of rules. First, scalability to the number of actions was tested by gradually increasing the number of actions. Second, scalability to the number of irrelevant constant sensors was tested by increasing the number of constant valued sensors while the robot executed a fixed sequence of 50 actions. Finally scalability to the number of irrelevant random sensors was performed while executing same sequence of actions defined in the second experiment.

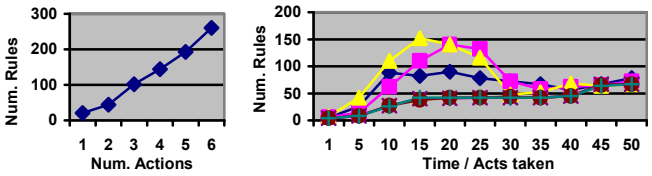

Fig. 5 (a) Scale actions (b) Scale constant & random sensors

As seen in Figure 5 (a), SBL showed linear growth with respect to the number of actions available (note that the grade depends on the ordering of the actions). In Figure 5 (b) the 3 lines at the bottom which almost overlap represent the responses for 1, 2 & 3 constant sensors. So, SBL can scale to an arbitrary number of constant or failed sensors. The 3 lines at the top represent the response for 1, 2 & 3 random sensors. Notice that at the onset the number of rules grow, but feature relevance eventually disregards these sensors and the rules are gradually forgotten, so the size of the learned model does not grow exponentially.

B. Fault Tolerance

Table 1: Details of SBL experiments

	Features available	*Unexpected fault introduced*	*Rules*		*Avg. actions*
			Max	*Avg*	
1	Exist, Size, Loc., Range	None	420	148	88
2	Exist, Size, Loc. Range, Const	Add irrelevant (constant) sensor	426	152	92
3	Exist, Size, Loc., Range, Const, Rnd	Add irrelevant (random) sensor	512	155	103
4	Exist, Size, Loc., Range, Const , Rnd	Rotate camera after a while	600	144	136
5	Exist, Size, Loc., Range, Const, Rnd	Switch left and right actions	740	139	152

Table 1 lists five experiments for fault tolerance with increasing complexity. The first experiment tested the minimal configuration of hardware required to learn the environment for successful navigation to a goal scene when no deliberate fault is introduced. The second and third experiments tested the scalability of learning to irrelevant sensors. The fourth experiment tested sensor failure and subsequent recovery. The

fifth experiment tested actuator failure and recovery. In all these experiments, SBL succeeded in learning to recover from the fault and accomplish goal tracking without any explosions in the size of the world model.

C. *Transfer Learned Knowledge across Goals*

An exercise similar to the Morris water maze [9] was used to test goal transfer in SBL. Two sets of experiments were carried out with 9 trials per experiment. First, the location of the hidden platform was fixed, but the starting location of the robot was changed after every 3 trials.

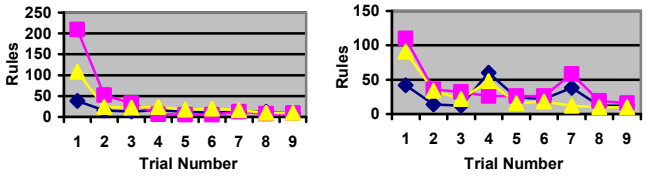

Fig.6 (a) Fixed goal, changing start (b) Fixed start, changing goal

Figure 6 (a) shows the number of actions executed during each trial to reach the goal for 3 separate experiments. It indicates that SBL is able to learn the location of the hidden platform as the number of executed actions decrease with more trails. Notice that the robot can use the learned model to reach the goal from any initial location.

In the second set of experiments the starting location was fixed, but the hidden platform (goal) was relocated after every 3 trials. The results for 3 experiments are shown in Figure 6 (b). Notice the peaks, which indicate that the robot was surprised and realized that the hidden platform has been relocated forcing it to search again. On subsequent trials it is able to track to the goal with fewer actions as it has learned the new location of the hidden platform by transferring all the learned knowledge which is still considered useful.

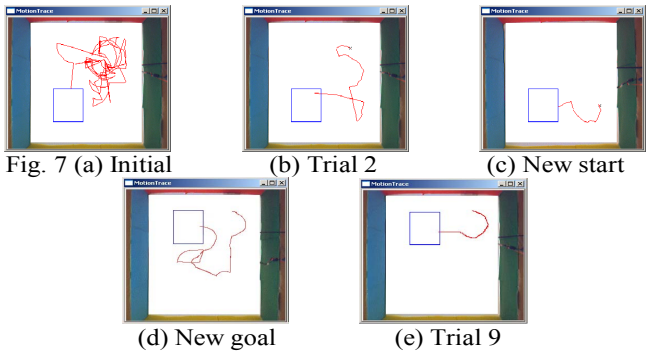

Fig. 7 (a) Initial (b) Trial 2 (c) New start

(d) New goal (e) Trial 9

Figure 7 shows the robot's moving traces in different trial. In 7(a), the robot explores the environment and learns a model for the task. In 7(b), the robot uses the learned model to get to the goal quickly. In 7(c), the robot is placed at a new starting location, but its learned model is still valid so that it can reach the goal quickly. In 7(d), the goal is secretly relocated, and the robot first goes back to the old goal location and was surprised that the goal is no longer there. It then starts exploration and finds the new goal location. In 7(e), the newly learned model is used to reach the new goal quickly. These experiments [10] showed that SBL can successfully solve the water maze with faster goal transfer and far less training than most model-less learning algorithms such as reinforcement learning [11].

VI. Conclusion & Future Work

This paper presents surprise-based learning that enables a robot to learn from an environment, adapt to unexpected changes in sensors and actions, and solve the water maze procedure autonomously, with no prior knowledge about the environment, its sensors, or the consequences of its actions. SBL proceeds by learning and relearning a model of the environment using prediction rules, where each rule describes the observation of the environment prior to the execution of an action, and forecasts the observation of the environment after its execution. The robot learns by investigating "surprises", which are inconsistencies between the predictions and the observed outcome. Therefore, each time a surprise occurs, SBL attempts to identify its cause and update the world model. This facilitates adaptation to changes in the environment. Rule rejection or the ability to detect and discard inappropriate rules, permits SBL to generate several prediction rules and quickly prune them in an effort to converge the model to an accurate representation of the environment. Similarly, feature relevance permits SBL to learn the sensor-actuator coupling autonomously. Hence, SBL is able to adapt to both changes in the environment and changes in the robot's sensors and actuators, making it adaptive and fault tolerant in these challenging situations.

In future, several improvements need to be made to SBL in order for it to be deployed in a large environment, including the ability to identify objects and learn concepts by grouping features. Probabilistic branching of prediction rules would be required to deal with ambiguity caused by features that are no longer unique. Several of these improvements are currently being tested.

References

[1] N. Ranasinghe., W-M. Shen, "*Surprise-Based Learning for Developmental Robotics"*, ECSIS symposium on Learning and Adaptive Behavior in Robotic Systems LAB-RS 08, Aug. 2008.

[2] W.-M. Shen, "*Complementary discrimination learning: a duality between generalization and discrimination*.", 8th National conf. on Artificial Intelligence, MIT Press, 1990.

[3] W.-M. Shen, "*Autonomous Learning From The Environment",* New York, W.H. Freeman and Company, 1994.

[4] X. Hurang, J. Weng, *"Novelty and reinforcement learning in the value system of developmental robots"*, 2nd Intl. Workshop on Epigenetic Robotics, 2002.

[5] N. Ranasinghe, W-M. Shen, *"The Surprise-Based Learning Algorithm"*, USC ISI internal publication, April 2008, ISI-TR-651.

[6] J. Bongard, V. Zykov, H. Lipson, *"Resilient machines through continuous self-modeling",* Science, Nov. 2006. 314: 1118-1121.

[7] M. Schembri, M. Mirolli, G. Baldassarre, "Evolving internal reinforcers for an intrinsically motivated reinforcement-learning robot", IEEE International Conference on Development and Learning, 2007.

[8] B. Salemi, M. Moll, W. Shen, "*SUPERBOT,* Intelligent Robots and Systems", Intl. conf. on Intelligent Robots and Systems, IROS, October 2006, pp.43-52.

[9] R. Morris, "*Developments of a water-maze procedure for studying spatial learning in the rat"*, Journal of Neuroscience Methods, May 1984, 11 (1): pp. 47–60.

[10] N. Ranasinghe, "Videos of *Surprise-Based Learning experiments"*, Available: http://www.isi.edu/robots/media-surprise.html

[11] E. Stone, M. Skubic, M. Keller, *"Adaptive temporal difference learning of spatial memory in the water maze task",* Intl. conf. on Development and Learning, ICDL 08, Aug. 2008.