The Surprise-Based Learning Algorithm

Nadeesha Ranasinghe, Wei-Min Shen

Abstract— **This paper presents a learning algorithm known as 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.**

Index Terms— **Autonomous robot, surprise-based learning, world model, plan, navigation, features, predict, complementary discrimination, reinforcement learning, developmental robotics.**

I. Introduction

THIS paper addresses the problem of a robot having to learn and navigate in an unknown environment with no prior knowledge about its actions or their impact on that environment so as to accomplish the task of reaching a particular scene. In particular, a modular mobile robot is tasked to navigate within a confined space consisting of several discernable features.

How should the world be modeled within the robot? What actions and sensors should a robot have to solve this problem? What action should the robot select when its goal is to reach a blue wall, but it currently sees a red wall? Should every feature be recorded before and after performing an action, or is it sufficient to record the most significant change? How can the robot identify the most significant change? Can the robot cope with sensing or actuation errors? These are a few important questions that identify some of the challenges of this task.

Since the robot has no prior knowledge it must gradually acquire some knowledge through a structured learning process in order to make any progress, as random motion combined with luck is unsatisfactory. This problem is interesting as it is motivated by human learning, which is a critical step in developmental learning. When placed in an unknown environment, a human child is able to interact with the environment and learn how to accomplish the task [1]. The bold ambition of most Artificial Intelligence and Robotics research is to achieve or exceed the intelligence demonstrated by a human, on a robot. A quick reflection makes it immediately visible that this is a very ambitious task as there are many unknowns in human learning, and straightforward replication of the technique simply does not exist.

We envision a scenario of learning as follows. First the robot will autonomously perform learning and problem solving inside a small static environment with a limited number of sensors. Then, as rigorous testing continues, constraints on the task, such as the number of sensors and actions will be varied to determine the competence of the approach. Hence, the problem addressed here is designed to be a building block on which an algorithmic solution will be devised and tested that could potentially be expandable to larger more complex environments or to other autonomous learning tasks.

The goal of learning is to create a world model as set of prediction rules, which provides both a model of the features in the environment as well as knowledge about the robot's actions that could be used to affect this environment. The proposed solution is an algorithm called "Surprise-based Learning" (SBL), which builds upon a learning paradigm called Complementary Discrimination Learning (CDL) [2-4] that uses discrimination as a means to accomplish both generalization and specialization. SBL extends CDL by overcoming many of its weaknesses to learn a world model based on a given set of percepts and internal motor commands. In this paper, the algorithm assumes the ability to (1) recognize and label colors which are features that are not predefined (dynamic labeling), (2) compare changes in features, and (3) perform a predefined set of "internal motor command sequences". In SBL such a sequence is referred to as an "action" even though the robot does not know what its consequences are. For example, an action called "action1" is given to robot as a sequence of motor commands, but the robot does not know if "action1" will cause itself to move or turn. The outcome of each action must be learned, just as a human learns to swim by "waving arms" and observing whether it caused the body to move forward or backward.

The proposed world model is a set of rules that predict the relationship between the observed features in the environment and the noticeable changes that occur when an action is performed by the robot. Typically a prediction rule consists of conditions, an action and predictions, in which conditions correspond to the observed state of the environment before an action is taken and the predictions forecast the observed state of the environment as a result of taking the action. A new rule is created by performing an "unknown" action and recording

Manuscript received April 11th, 2008. This work is supported by the U.S. Air Force Office of Scientific Research, Grant #FA9550-06-0336.

Wei-Min Shen (e-mail: shen@isi..edu) & Nadeesha Ranasinghe (e-mail: nadeesha@isi.edu) are with the University Of Southern California's Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina Del Rey, CA 90292, USA.

the most noticeable change as its prediction. This rule is then maintained by repeating the associated action and recording new information when an unexpected outcome or "surprise" occurs. While learning, the world model can be queried to form a plan, which is a sequence of actions executed by the robot to accomplish a given task.

The rest of this paper is organized as follows. Section II discusses related work and the motivation for selecting this approach. Section III describes the proposed framework, while section IV describes the logic used to create and maintain the world model. Section V describes the experimental setup to test and demonstrate learning, while section VI explains implementation of the surprised-based learning algorithm in detail. Section VII discusses some of the experimental results. Finally, section VIII summarizes the contribution of this research and outline directions of future research.

II. Related Work

Driven by the desire to achieve complete autonomy several decades of artificial intelligence and robotics research has produced many learning techniques, but the problem of autonomous learning still remains largely unsolved. At present most learning algorithms can be classified as supervised, unsupervised or reinforcement learning [5]. Supervised learning (SL) requires the use of an external supervisor that may not present here. In contrast, unsupervised learning (UL) opts to learn without any feedback from the environment by attempting to remap its inputs to outputs, using techniques such as clustering. Hence, it may overlook the fact that feedback from the environment may provide critical information for learning.

Reinforcement learning (RL) receives feedback from the environment. Some of the more successful RL algorithms used in related robotic problems include Evolutionary Robotics [6] and Intrinsically Motivated Reinforcement Learning [7]. However, most RL 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.

Complementary Discrimination Learning (CDL) [8] 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 activities, such as abstraction, surprise analysis with noisy sensors, dynamic goal assignment, prediction creation and continuous maintenance, are necessary for robotic learning.

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] proved that an internal model such as an action or sensor model of a robot can be learned without prior knowledge given that the environment and the robot can be reset to its initial configuration prior to each experiment, which is not supported in this problem.

Another promising approach is Intrinsically Motivated Reinforcement Learning (IMRL) where the robot uses a self-generated reward to learn a useful set of skills. IMRL has successfully demonstrated learning useful behaviors [11], and a merger with ER as in [12] was able to demonstrate navigation in a simulated environment. The authors have mentioned that IMRL can cope with situated learning, but the high dimensional continuous space problem that embodiment produces, is beyond its 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. In the autonomous robotic learning problem we are interested in, the learner must accommodate limited processing, noisy actuation and limited or noisy sensing available on a physical robot. Furthermore, the ability to predict and be "surprised" by any ill effects is also critical. This powerful learning paradigm has been analyzed, theorized, explored and used in a few other learning applications such as traffic control [15] and computer vision [16].

Alternatively, learning a world model could be accomplished to some extent by simply recording the environment using conventional map building such as Simultaneous Localization and Mapping (SLAM) [17]. However, accurate-enough models for the robot's actions and sensors must be given to facilitate this type of learning. These models are not available for developmental learning, which is motivated by previous researches presented in [18-21].

III. The Surprise-Based Learning Framework

The dictionary defines "surprise" as "a sudden or the unexpected encounter". In 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 VI discusses how a surprise is encountered and handled by SBL. The algorithm follows the framework visualized in Fig. 1.

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.

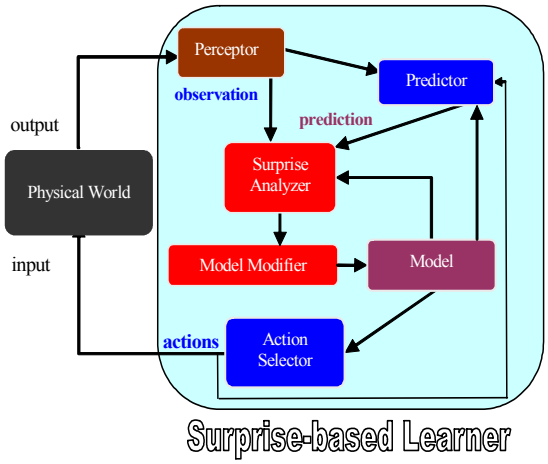

Fig. 1: Surprise-based Learning Framework

IV. Prediction Model

In surprised-based learning, knowledge concerning the relationship between the robot's action and its impact on the environment is captured within a prediction rule. Therefore, the model of the physical environment is represented as a set of rules, where each rule is a triple comprised of conditions, an action and corresponding predictions as in (1).

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

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

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

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. Condition1 ≡ (feature1, >, 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 ≡ (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 in section VI. *D*.

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.

To make a learning algorithm feasible for a real robot, the knowledge representation must be extremely compact and efficient. Thus, it is not only important that the rules should be easy to parse; the number of rules should not increase infinitely over a period of time. This means that the model should be rich enough to capture sufficient information to accomplish any given task, yet abstract enough to record and query the entire physical environment, while being compact enough for accomplishing the learning within a feasible amount of time.

V. Experimental Setup

The current implementation consists of, a single SuperBot [22] modular robot 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 approximately 10".

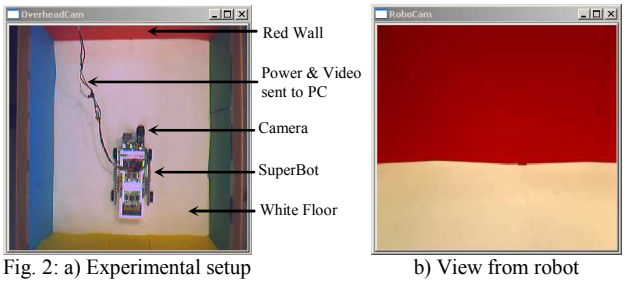

Fig. 2: a) Experimental setup b) View from robot

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 [23] 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’).

With this environment, we plan to demonstrate SBL in 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 to the robot, 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, the above tasks will 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.

VI. The Surprise-Based Learning Algorithm

The SBL algorithm follows a cycle which includes rule creation, selection, validation, maintenance via rule splitting & rule refinement, rule abstraction and planning. The general procedure is outlined as follows:

The cycle starts by selecting an action either randomly or via a plan. Then, rule selection returns all prediction rules whose conditions match the current state of the environment, should one or more exist. The action is executed. If a matching prediction rule did not exist, a new rule is created through rule creation by comparing the current state of the environment against its previous state. However, if there were matching prediction rules, then each rule is assessed individually as described below. First, rule validation ascertains that the current state of the environment matches its predictions. If all predictions are true, then the last successful instance of the rule is updated, and the cycle continues. Else, there is a surprise, thus rule maintenance is invoked to perform rule splitting or rule refinement depending on whether the selected rule already belongs to a pair of complementary rules that have been split previously. If rule splitting is performed, then a new complementary rule is generated with new predictions and the selected rule is updated with new conditions and possibly new predictions. However, if refinement is performed, then both complementary rules are updated with new conditions but the predictions remain unaltered. Once rule maintenance is complete the cycle restarts such that an action can be selected via planning. Optionally, rule abstraction can be performed during rule splitting and refinement so as to aid planning.

Pseudo code for SBL is presented in Algorithm 1. Subsection A discusses surprise analysis, while subsection B describes the logic used to create split and refine rules. Subsections C through F delves into the intricacies of rule creation, selection & validation, splitting and refinement respectively. Finally, subsection F describes the planner together with rule abstraction.

Algorithm 1: Surprise-based Learning

```
1: Observe the environment via sensors
2:
3: while (ALIVE) do
4:     if (a goal scene is given) then
5:         if (plan is NULL) then
6:             Find a plan as a sequence of rules that will lead to
               the goal { see subsection F }
7:         end if
8:
9:         Action = Next action from current plan
10:    else
11:        Action = Choose a random action
12:    end if
13:
14:    predictionRules = Match the current observations and
       select the matching prediction rules which forecast the
       outcome of this action { see subsection C }
15:
16:    Perform(Action)
17:    Observe the environment via sensors

18:    if (predictionRules is NULL) then
19:        CreateNewRule() { see subsection B }
20:    else
21:        for (each Rule in predictionRules) do
```

```
22:         if (The outcome matches the prediction) then
23:             Update the last success of the predicted rule
24:         else // this is a "surprise"
25:             if (the rule has never been split before) then
26:                 SplitRule(prediction) { see subsection D }
27:             else
28:                 RefineRule(prediction) { see subsection E }
29:         end if
30:     end for
31:   end if
32:
33: end while
```

A. Surprise Analysis

Humans are able to gather data from multiple sensors and determine which sensor or combination of sensors is required to easily perform certain tasks. Similarly, a robot with many sensors must determine the stimulus or stimuli that are most strongly associated with change, given two or more observations where multiple stimuli are present simultaneously. This is known as surprise analysis, which attempts to identify the most likely reason for a surprise. In other words it is the most noticeable or most significant change.

This is a difficult process as a prediction could contain one or more features and these features could have complex relations to other features. It 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, as is visible in the results presented in section VII.

B. SBL Logic

Rule Creation:

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

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

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

After splitting, the 2 rules are marked as "complementary".

Rule Refinement, given Rule 1.1 failed:

(C_3 being the reason for surprise)

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

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

Rule Refinement, given Rule 1.2 failed:

(C_3 being the reason for surprise)

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

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

C. 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.

For example consider executing the forward action when the robot is located as in Fig. 2a), which results in Fig. 3b). 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.

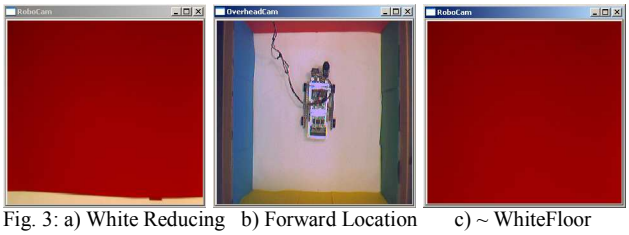

Fig. 3: a) White Reducing b) Forward Location c) ~ WhiteFloor

Rule 1: (RedWall, %, 0) → FORWARD → (RedWall, >)
Rule 2: (RedWall, %, 0) → BACKWARD → (RedWall, <)

D. 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. 3b). 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.

E. 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) → FORWARD → (RedWall, >) OR NOT (WhiteFloor, ~)

Rule 3: (RedWall, %, 0) AND NOT (WhiteFloor, %, 0) → FORWARD → (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) → BACKWARD → (RedWall, <)

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

F. 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) → BACKWARD → (RedWall, <)

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

G. 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.

VII. Experiment Results & Analysis

Eight sets of experiments were conducted to establish the feasibility of SBL in solving this problem, to demonstrate that the environment was sufficiently complicated 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:

http://www.isi.edu/robots/media-surprise.html

Table 1: Components of each set of SBL 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

A. Experiment Set A – Most successful parameters

These experiments test the most successful implementation. The 6 actions were forward, backward, small left (turn angle close to -30º), large left (turn angle close to -45º), small right (turn angle close to 30º) and large right (turn angle close to 45º). Regardless of when the goal was assigned, the robot always managed to reach it, either by following a plan or by randomly encountering the goal. Hence, the following criteria were highlighted for successful learning.

- The total number of surprises should reach a constant after the robot has traversed the environment several times, which means that actions produce little or no surprises as learning progresses.
- The planner is able to generate a feasible plan from the current state to the goal state without any random actions, indicating that the necessary feature relations have been learned.

Given the criteria above, any experiments that reached the goal without a plan were discarded from the results and repeated as this would introduce false positives. Hence, the actual number of experiments is larger than the amount recorded in the tables. Also, assigning the goal prior to commencing learning would almost always lead to this result, thus it was not considered in subsequent experiments.

The results indicated that SBL was able to learn accurate prediction rules over a period of time, resulting in surprises receding and prediction successes continuously increasing. A successful trace of SBL is in Fig. 4. A few experiments were carried out by toggling the actions (such as switching forward and 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. In Table 2, the 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, even this experiment was able to seek the goal eventually.

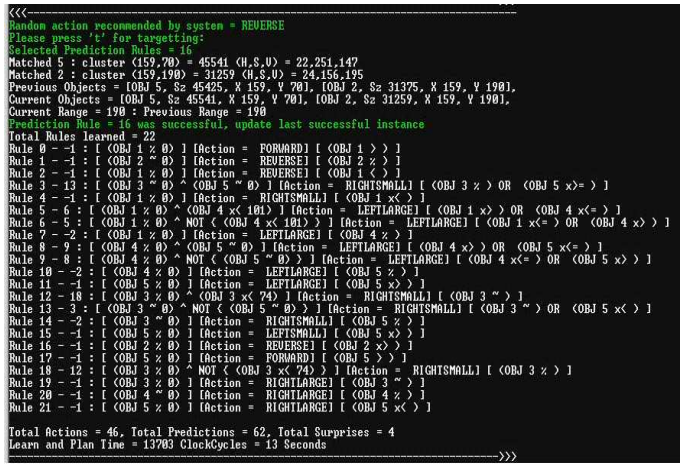

Fig. 4: SBL trace containing 22 prediction rules

B. *Experiment Set B – Changing the number of actions*

The number of actions was reduced to 4, namely forward, backward, left and right. At times, when the planner was invoked after traversing the environment thoroughly SBL was able to generate a successful plan. However, even without any surprises, there were times where SBL was unable to generate a successful plan. The root cause was the size of motion for each action affecting rule splitting.

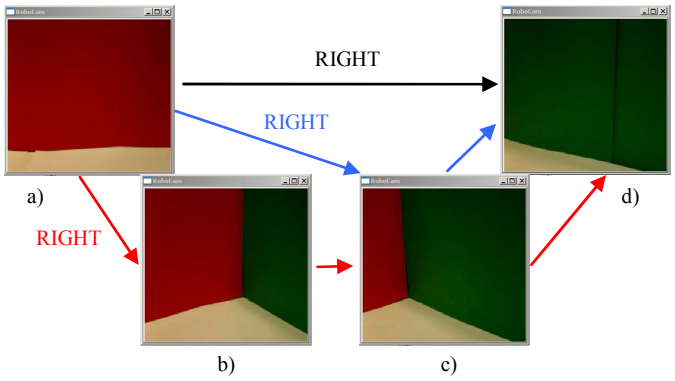

Fig. 5: Scenarios that could be encountered when performing a turn action

Rule 6: (RedWall, %, 0) → RIGHT → (RedWall, x<)

Rule 7: (RedWall, %, 0) AND NOT (RedWall, x>, Value3) → RIGHT → (RedWall, x>=)

Rule 8: (RedWall, %, 0) AND NOT (GreenWall, ~, 0) → RIGHT → (RedWall, x>=) AND (GreenWall, x<, 0)

Rule 9: (RedWall, %, 0) → RIGHT → (GreenWall, x<, 0)

Consider rule splitting using "before past" and "before current" instances encountered while performing the right action in Fig. 5. Given Rule 6, note that the there are three possible scenarios which could split the rule and result in Rule 7 (a→d or a→b→c→d) or Rule 8 (a→c→d) depending on the magnitude of the turn. Rule abstraction using Rule 8 generates Rule 9, which is an important feature transition that is missing in the complementary pair containing Rule 7. The planner needs to capture feature relations such as this corner transition in order to generate a valid plan. Therefore, the number of discrete actions is an important parameter in SBL. Increasing the number of actions resulted in an increase in the number of prediction rules as well as an increase in the time taken for surprises to recede. However, the quality of the world model improved sufficiently well enough to generate valid plans as seen in experiment set A.

C. *Experiment Set C – Changing the operator priority*

In this set of experiments the displacement of the center of mass of each feature was considered only if there was no noticeable change in the size of the feature. The objective was to determine the impact of changing the order or priority of surprise analysis. Consider the rules created when turning right in Fig. 6. As was expected the >, <, = took precedence over x>, x< resulting in these operators becoming redundant. These experiments produced Rule 10 and Rule 11 that resulted in many surprises in contrast to the experiments in set B, where the operator priority was toggled to create the rule 12. The surprises were due to the ambiguity caused by considering visual qualitative information during turning instead of the displacement of the center of mass. Testing indicated that the use of the horizontal axis with the proper priority was sufficient to satisfy the success criteria SBL, as the vertical axis (y>, y<) was redundant when the feature size was considered. This concludes that the operator priority is an important parameter in SBL.

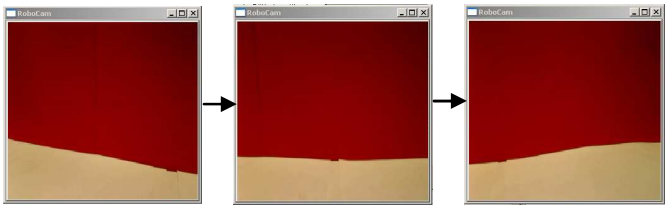

Fig. 6: Feature size comparison when performing a turn action

Rule 10: (RedWall, %, 0) AND (RedWall, <, Value4) → RIGHT → (RedWall, >)

Rule 11: (RedWall, %, 0) AND NOT (RedWall, <, Value4) → RIGHT → (RedWall, >=)

Rule 12: (RedWall, %, 0) → RIGHT → (RedWall, x<)

D. *Experiment Set D – Changing the number of operators*

The vision sensor data stream contains a large amount of potentially useful information. If the number of comparison operators is changed surprise analysis is affected, meaning that SBL's ability to identify the exact cause of a surprise is altered. This could result in the same inaccurate predictions as seen in experiment set C, and just as before SBL may not be able to find or follow a plan to the goal without a surprise interrupting the execution.

E. Experiment Set E – Learning from human influence

The idea was to apply a developmental robotic technique such as running SBL with the guidance of a human operator, in an attempt to identify how a human would solve this problem. For this purpose, the robot's motion was disabled, allowing a human operator to move the robot manually, and the next action would be confirmed or overridden by this user. Two human operators with no prior experience or knowledge of the algorithm conducted two experiments each. There were some similarities and differences in the way both users conducted the experiments.

An important similarity was that both users chose to ignore random actions and proceeded to perform a particular action two or three times, allowing them to validate the SBL prediction rules. This bias towards repeatedly selecting the same action until the surprises subside helps construct more general rules than rules generated by purely random actions which tend to be more specific. In all previous experiments random actions were biased to increase the probability of learning a prediction rule and immediately testing it, hence this set of experiments validated this bias.

A major difference between the two humans was their choice in manual motion, namely the distance traveled forward & backward, and the angle turned left & right. One user used relatively larger motions than the other, leading to two qualitatively different models of the environment. The differences confirmed the root cause identified in experiment set B.

F. Experiment Set F – Removing human influence

When the bias was removed from the action selection criteria, SBL chose completely random actions to execute. SBL had to perform a large number of actions for a plan to be generated, as the number of surprises was much larger than with the biased action. Even with a plan the system encountered many surprises prior to reaching the goal as the rules reflected the erratic motion encountered during learning.

G. Experiment Set G – Changing the number of sensors

The range sensor was removed to determine the impact caused by changing the number of sensors. As long as the sensor provides data that could potentially disambiguate the cause for a surprise its availability impacts the quality of the world model. Consider performing the backward action in Fig. 3c) given in section VI. *E*, which could result in either Fig. 3c) again or Fig. 3a) depending on the robot's proximity to the wall as stated before. Unfortunately, without the range sensor SBL loses its ability to predict which scenario would occur next. Therefore, the number of sensors is a vital parameter.

H. Experiment Set H – Minimal amount of parameters

This is possibly the most minimal configuration of SBL. As previous sets of experiments suggest the lack of sensors and features comparison operators completely hinders its ability to learn and plan in this environment, resulting in the goal being unattainable based on the success criteria.

I. General observations from the experiments

In addition to the results discussed earlier, another important observation worth noting is 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 that due to the complementary nature of rule splitting, a specialized rule has very precise predictions, while its complementary generalized rule could have some potential to be explored further. If the predictions of a rule contained only ~, %, =, >= or <= there is potential to explore further. During rule selection such prediction rules are flagged, for the purpose of forcing rule creation after rule validation is performed. In addition, all newly created rules are compared against the stored prediction rules so as to avoid adding multiple occurrences of the same rule into the world model.

From the results, it is evident that the amount of information presented to SBL defines the quality of the world model and consequently the plan. Changing the number of actions, surprise priority, the number of comparison operators, biased actions, and the number of sensors directly affects the learning time and the quality of the world model.

VIII. Conclusion & Future Work

This paper presents a new learning algorithm called surprise-based learning that enables a robot to reach a goal by navigating an unknown environment with no prior knowledge about its actions or their impact on that environment. The SBL algorithm models the world as a collection of prediction rules, which forecast the outcome of performing an action. 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 a perfect 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 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.

ACKNOWLEDGMENT

The authors would like to thank the U.S. Air Force Office of Scientific Research. We are grateful to Rizwan Khan and Feili Hou for conducting the human driven experiments and all the members of the Polymorphic Robotics Laboratory at the University of Southern California's Information Sciences Institute for their feedback and support.

REFERENCES

[1] Koslowski, B., J. Bruner, "Learning to use a lever", *Child Development*, 43:790-799, 1972.

[2] Shen, W.-M., Simon H., "Rule creation and rule learning through environmental exploration", Eleventh Joint Conference on Artificial Intelligence, Morgan Kaufmann. 1989.

[3] Shen, W.-M., "Complementary discrimination learning: a duality between generalization and discrimination", *Eighth National Conference on Artificial Intelligence*, MIT Press. 1990.

[4] Shen W.-M., "Discovery as Autonomous Learning from the Environment," Machine Learning Journal, vol. 12, Aug. 1993, pp. 143–165.

[5] Kaelbling L., Littman M., Moore A., "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research, Vol. 4, 1996.

[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, Vol. 11, 2007.

[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, 92: 169-229, 1997.

[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, Oct 2004.

[17] Thrun, S., "Robotic Mapping: A Survey", Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

[18] Piaget J., "The Origins of Intelligence in the Child", Norton, 1952.

[19] Simon H., Lea G., "Problem solving and rule induction: A unified view", Knowledge and Cognition, Hillsdale, NJ, 1974.

[20] Cohen P., Atkin M., Oates T., Beal C., "Neo: Learning conceptual knowledge by sensorimotor interaction with an environment", Intl. Conference on Intelligent Agents, 1997.

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

[22] 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, Oct. 2006.

[23] Comaniciu D., Meer P., "Mean Shift: A Robust Approach toward Feature Space Analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, May 2002, pp. 603-619.