Autonomous Adaptation to Simultaneous Unexpected Changes in Modular Robots

Nadeesha Ranasinghe
Information Science's Institute
University of Southern California
Marina Del Rey, USA
nadeesha@isi.edu

Wei-Min Shen
Information Science's Institute
University of Southern California
Marina Del Rey, USA
shen@isi.edu

***Abstract*— To accomplish tasks in the real world, a robot (especially a modular and self-reconfigurable one) must be able to autonomously react and adapt to unexpected and possibly simultaneous changes to its self (such as its sensors, gaits, goals and modules) and dynamic situations/configurations in the environment. Failures, faults and reconfiguration that commonly occur in modular robots result in simultaneous changes that are at times unexpected as well as unpredictable. This paper presents an integrated technique called Surprise-Based Learning (SBL), which aims to enable a reconfigurable robot to autonomously adapt to changes. The main idea is to have a robot perform both learning and representation in parallel by constructing and maintaining a *predictive model* about itself and the environment.**

Modular reconfigurable robots, developmental robotics, learning, unexpected changes, fault-tolerance, failure-tolerance, surprises, adaptation.

I. INTRODUCTION

No matter how carefully a robot is engineered, its initial knowledge is bound to be incomplete or incorrect with respect to the richness of the real world. Thus, an autonomous robot must deal with dynamic situations caused by changes in its sensors, gaits and goals, as well as changes in the environment. This is especially important for modular and self-reconfigurable robots because such robots have built-in capability to change themselves and they must have effective mechanism to decide when, what and how to change. This paper provides a first step towards that direction.

To start this investigation, we define for now *unexpected changes* as addition and deletion of sensors, gaits, goals and alterations in the robot's and environment's configuration. Typically, in a modular robot the addition or removal of modules results in changes to the robot's sensors and gaits. On top of this, sensors and gaits may also have unexpected changes in their "meaning" or "definitions". A *definition-change* in sensors means that the sensor response has changed, e.g. a module with a camera being rotated. A definition-change in gaits means that the actuator response has changed, e.g. flipping a butterfly gait upside down results in it moving in the opposite direction.

There are several major challenges for this problem. First, unexpected changes in modular robots occur simultaneously in sensors, gaits, goals or other related aspects. Thus, a robot cannot assume any particular component to be fault-free, or guaranteed redundancy. This is particularly true for heterogeneous robots. Second, due to the lack of permanently correct (predictive) models for any robot or environment, a robot must constantly check and refine its models. The detection of change may not be under the supervision of external guidance, and on-board resources may exclude expensive solutions. In most cases, the robot must cope with continuous (non-discrete), uncertain and vast information space.

Our approach to this challenging problem is to enable the robot to perform simultaneous representation and learning. The main idea is for the robot to continuously detect changes and adapt its representation for the proper interactions between itself and the environment while achieving its given goals. This requires a coherent and flexible integration of perception, planning, memory, reasoning, learning, focusing of attention, and discovery. Specifically, we use a technique called Surprise-Based Learning (SBL) as a framework developed to address these challenges.

SBL attempts to create a *predictive model* of the environment based on the robot's interactions with the environment. When a gait is selected towards accomplishing a goal, the predictive model returns the predicted consequences of executing those actions, and if it does not match the actual observation received after the execution of the gait/actions we call it a "surprise". The algorithm attempts to identify the cause of the surprise and update its model accordingly. Hence, a surprise essentially detects an unexpected change with respect to the existing model. We assume that the robot has sufficient capability to compensate for the changes and update the model.

The rest of this paper is organized as follows. Section II highlights related work. Section III clarifies the terminology used in this paper. Section IV provides details of the core approach, while section V explains the specifics on handling unexpected changes. Experimental results are presented in section VI. Section VII provides some discussion for future research in this direction.

II. RELATED WORK

Typically there are 3 attitudes towards dealing with unexpected changes in robots as identified by Saffiotti [1]. The 1st is to get rid of it through precise engineering. The 2nd is to

tolerate it by utilizing redundancy with carefully designed contingency routines. The 3rd is to reason about it by using techniques for the representation and manipulation of uncertain information.

In the context of modular robots the attitude of precise engineering does not prevent unexpected changes that occur as a result of reconfiguration. As per the attitude to tolerate unexpected changes in sensors and actions the robot must be able to detect the change and then handle it. For this purpose one feasible approach is to provide models of the sensors, actions and environments, such that when an exception occurs it could be handled by pre-prepared contingency strategies. Unfortunately, modular robots are developed to operate in environments and handle tasks that may not be known a priori. Therefore it is not always possible to acquire accurate models of sensors [2], actions [3-6] or environments [7] as demonstrated on several non-modular robots.

The attitude towards reasoning and manipulation of uncertainty, known as data-driven adaptation aims to overcome the shortcomings of the previous attitudes. Probabilistic sensor fusion [8-9], simulating and reasoning about uncertainties [10], modeling and re-planning [11] are several techniques that have yielded some results on non-modular robots. Most of these techniques rely on either the use of controllable environments i.e. the ability to reset or alter accordingly, or the availability of one or more models to verify the other i.e. use the sensor model to establish that an actuator has failed etc., which may not be possible with modular robots.

Despite numerous successes, none of these techniques attempt to deal with simultaneous unexpected changes in all aspects, including changes in sensors, actions, goals and the robot's and environment's configurations, in an unsupervised manner. To our knowledge, very little work has been done in modular robotics to reason about and deal with surprises that arise due to the special capabilities of reconfigurable robots. The SBL technique [12] is a first attempt towards this direction. In this paper, we limit the scope of a robot's actions to be gaits only, and changes in configuration to be environmental only (the changes in robot's configuration will be considered in future papers).

III. Basic Definitions

A *gait* g_i is a set of predefined actuator commands over a period of time. A set of gaits $\{g_1, g_2, \dots, g_n\}$ together with their applicable configurations is given, e.g. "caterpillar", "sidewinder", "butterfly" etc. A *sensor* s_i returns a set of

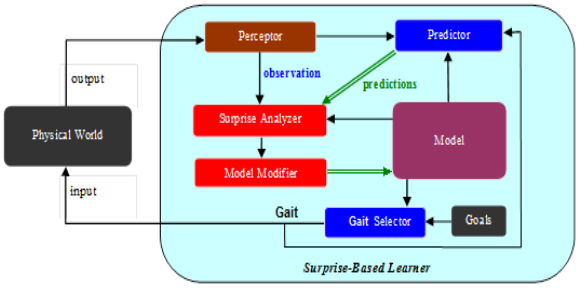

Fig. 1. Surprise-Based Learning framework

entities $\{e_1, \dots, e_f\}$. An *entity* e_i is defined as set of attributes $\{a_1, \dots, a_g\}$. An *attribute* a_i has either a continuous value [p, q] or a set of discrete values $\{v_1, v_2, \dots, v_h\}$. A set of sensor, entity and attribute mappings is given. An *operator* or comparison operator is a mechanism that enables the learner to reason about an entity, or an attribute's values. A set of operators $\{\odot_1, \odot_2, \dots, \odot_n\}$ is given, $\odot$ may be presence(%), absence(~), increase(↑), decrease(↓), greater-than(>), less-than(<), equal(=), etc.

An *observation* at time t is a set of all sensor readings obtained $O_t = \{[s_1e_1a_1] = v_1, \dots, [s_ne_ia_j] = v_k\}$. When a prediction rule is created at time t (after a gait has been executed), this observation is recorded as the "*base-result*", and observation made at time t-1 (before the gait execution) is recorded as the "*base-condition*". A *goal* is an observation that the robot wants to receive from the environment. A goal $G = \{[s_1e_1a_1] = v_1, [s_2e_1a_x] = *, \dots, [s_ie_ja_k] = v_l\}$. A set of goals is given. A goal is achieved at time t when it is contained in the observation at that time, i.e. $G \subseteq O_t$.

A *prediction* for time t is a forecasted observation made before time t. A prediction $P_t = \{[s_1e_1a_1] \odot v_1, [s_2e_1a_1] \odot v_2 \vee \dots \vee [s_2e_1a_1] \odot v_k, \dots, [s_ie_ja_k] \odot v_l\}$. A prediction is true at time t when $O_t \supseteq P_t$. A *surprise* occurs whenever there is a discrepancy between a prediction and its corresponding observation at time t. Currently we define that a surprise is true when none of the predicted values in a clause are contained within the observation, i.e. $P_t \backslash O_t \neq \emptyset$. The observation made at time t which resulted in a surprise, is called the "*surprised-result*", similarly the observation made at time t-1 is called the "*surprised-condition*".

IV. Surprise-Based Learning

A. Architecture

The framework for Surprise-Based Learning depicted in Fig 1 shows how the main modules of the system interact. The learned knowledge is represented in a model, which consists of prediction rules. The learning process highlighted in Fig 2 is as follows: i) A gait is randomly selected based on the current configuration or planned based on the current model. ii) The predictor returns all prediction rules in the model whose gait and conditions match the current observation as well as the selected gait. iii) The gait is executed. iv) A new observation is made. v) If no prediction was made or new learning opportunities exist, then new rules are created (see *Rule Creation*). vi) If surprises were detected (see *Surprise*

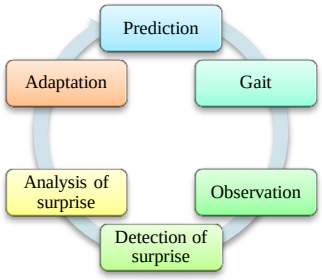

Fig. 2. Surprise-Based Learning process

Detection & Analysis) the model is revised (see *Rule Maintenance*) to reflect the new observation. This learning process is an enhanced version of Complementary Discrimination Learning [13], and it has been improved to handle continuous data, capture interrelations in the predictions and most importantly evaluate all possible causes of surprise in parallel.

$$\text{Rule} \equiv \text{Conditions} \rightarrow \text{Gait}^{+} \rightarrow \text{Predictions} \quad (1)$$
$$\text{Condition} \equiv (\text{Entity, Attribute, Comparison Operator, Value}) \quad (2)$$
$$\text{Prediction} \equiv (\text{Entity, Attribute, Expected Change, Value}) \quad (3)$$

Each prediction rule is a triplet as in (1) of "condition", "gait", and "prediction". Conditions are logical statements describing the state of observed entities and attributes prior to the execution of a specific gait one or more times as indicated by $^{+}$. A condition can be represented as a 4-tuple as in (2). Using the comparison operators given to the learner, each condition describes a relationship formed of an entity, its attribute and value. For example, the expression $\text{Condition}_1 \equiv (\text{entity}_1, \text{attribute}_1, >, \text{value}_1)$ means that Condition_1 is true if attribute_1 of entity_1 is greater than value_1. Several logically related conditions can be grouped together to form a sentence using 'And' and 'Not' logical operators. Predictions are sentences that describe the expected change in the state of observed entities and attributes as a result of executing a specific gait possibly a number of times. As seen in (3) a prediction can be represented using a 4-tuple. i.e. $\text{Prediction}_1 \equiv (\text{entity}_1, \text{attribute}_1, \uparrow, \text{value}_1)$ means that if the rule is successful the value of entity_1's attribute_1 will increase by the amount indicated in value_1.

An important aspect of prediction rules is that they can be sequenced to form a prediction sequence $[o_0, g_1, p_1, \ldots, g_n, p_n]$ where o_0 is the current observation at the current state, and g_i, $1 \le i \le n$, are gaits, p_i $1 \le i \le n$, are predictions. As the gaits in this sequence are executed, the environmental states are perceived in a sequence $o_1, o_2, \ldots, o_n$. Prediction sequences can be used to represent many concepts such as a plan, exploration, example, and advice in a unified fashion.

B. Rule Creation

New rules are created and added to the model by comparing the base-condition 'BC' and base-result 'BR' of executing a gait 'g' one or more times with the set of operators $\{\%, \sim, \uparrow, \downarrow\}$. The following functions return new prediction rules:

For each entity e in BC but not in BR, create $[\%e \xrightarrow{-g^{+}} \sim e]$; (4)
For each entity e not in BC but in BR, create $[\sim e \xrightarrow{-g^{+}} \%e]$; (5)
For each entity e_1 in BC not in BR and each e_2 not in BC but in BR, create $[\%e_1 \xrightarrow{-g^{+}} \%e_2]$; (6)
For each entity e in BC and BR, do
  for value v increased, create $[e.a \xrightarrow{-g^{+}} e.a\uparrow v]$; (7)
  for value v decreased, create $[e.a \xrightarrow{-g^{+}} e.a\downarrow v]$; (8)

Rule creation is invoked when the robot i) has no predictions for the selected gait, or ii) none of the predicted rules forecasted a change in the current observation, or iii) all predicted rules have been rejected after rule analysis.

C. Surprise Detection & Analysis

When the conditions of a prediction rule 'R' are satisfied by the current observation the rule's predictions are evaluated after the gait is executed. A surprise is detected if a prediction fails to be realized, i.e. the forecasted value was not observed.

The objective of surprise analysis is to identify the possible cause(s) of the surprise by comparing entities and attributes in the base-condition 'BC' with those in the surprised-condition 'SC' using the given set of comparison operators (typical set $\{\%, \sim, >, <, =, !=\}$). The following functions perform surprise analysis, which return a set of possible causes $[c_1, \ldots, c_z]$, where each cause is an expression of an entity or an attribute that was true in the base-condition but false in the surprised-condition:

For each entity e in BC but not in SC, cause (%e); (9)
For each entity e in SC but not in BC, cause (~e); (10)
If the values in the domain are ordered, then for each entity e in BC and SC, do
  for value $e.a.v_{BC} < e.a.v_{SC}$, cause $(e.a < e.a.v_{SC})$; (11)
  for value $e.a.v_{BC} > e.a.v_{SC}$, cause $(e.a > e.a.v_{SC})$; (12)
If the values in the domain are unordered, then for each entity e in BC and SC, do
  for value $e.a.v_{BC} != e.a.v_{SC}$, cause $(e.a != e.a.v_{SC})$; (13)

D. Rule Maintenance

Rule maintenance deals with updating existing rules when they are surprised by new observations in the surprised-result 'SR'. When a single rule R_0: $c_0 \xrightarrow{-g^{+}} p_0$ is surprised for the first time, a set of possible causes $[c_1, \ldots, c_z]$ is acquired from surprise analysis and used to perform rule *splitting* as follows:

For each cause c_x identify the expected change p_x as follows,
  For each entity e in SC not in SR, $(p_x = \sim e)$; (14)
  For each entity e in SR not in SC, $(p_x = \%e)$; (15)
  If the values in the domain are ordered, then for each entity e in SC and SR, do
    for value v increased, create $(p_x = e.a\uparrow v)$; (16)
    for value v decreased, create $(p_x = e.a\downarrow v)$; (17)
  If the values in the domain are unordered, then for each entity e in SC and SR, do
    for value v changed, create $(p_x = e.a.v_{SR})$; (18)
For each cause c_x create a complementary pair of rules as follows,
  Specialization R_{00}: $c_0 \wedge c_x \xrightarrow{-g^{+}} p_0 \vee \sim p_x$ (19)
  Generalization R_{01}: $c_0 \wedge \sim c_x \xrightarrow{-g^{+}} \sim p_0 \wedge p_x$ (20)

Hence, for each possible cause, rule splitting results in a new pair of complementary prediction rules that reflect both the original result p_0 and the surprised result p_x.

For any subsequent surprise that occurs in a pair of complementary rules R_A: $c_0 \wedge c_x \xrightarrow{-g^{+}} p_z$ and R_B: $c_0 \wedge \sim c_x \xrightarrow{-g^{+}} \sim p_z$, rule *refinement* is performed as follows:

For each cause returned by surprise analysis if it is has never been added to the surprised rule then it can be added safely as it is a new cause. Otherwise, it should not be added as the condition of the rule may grow indefinitely when the true cause is hidden.
When R_A is surprised, for each new cause c_y that is not specified in c_0 or c_x, do
  R_A : $c_0 \wedge c_x \wedge c_y \xrightarrow{-g^{+}} p_z$ (21)
  R_B : $c_0 \wedge \sim(c_x \wedge c_y) \xrightarrow{-g^{+}} \sim p_z$ (22)

When R_B is surprised, for each new cause c_y that is not specified in c_0 or c_x, do

$R_B : c_0 \wedge \sim c_x \wedge c_y \xrightarrow{-g^+} \sim p_z$ (23)

$R_A : c_0 \wedge \sim(\sim c_x \wedge c_y) \xrightarrow{-g^+} p_z$ (24)

When R_A is surprised, if a new cause cannot be identified then add the entire observation ($e_1, \ldots, e.a.v_n$),

$R_A : c_0 \wedge c_x \wedge \sim(e_1 \wedge \ldots \wedge e.a.v_n) \xrightarrow{-g^+} p_z$ (25)

$R_B : c_0 \wedge \sim(c_x \wedge \sim(e_1 \wedge \ldots \wedge e.a.v_n)) \xrightarrow{-g^+} \sim p_z$ (26)

When R_B is surprised, if a new cause cannot be identified then add the entire observation ($e_1, \ldots, e.a.v_n$),

$R_B : c_0 \wedge \sim c_x \wedge \sim(e_1 \wedge \ldots \wedge e.a.v_n) \xrightarrow{-g^+} \sim p_z$ (27)

$R_A : c_0 \wedge \sim(\sim c_x \wedge \sim(e_1 \wedge \ldots \wedge e.a.v_n)) \xrightarrow{-g^+} p_z$ (28)

Therefore rule maintenance keeps the prediction model accurate by performing rule splitting first followed by rule refinement every time a surprise occurs.

V. Handling Unexpected Changes

A. Goal Changes

SBL can use any standard planner on the prediction model to find a sequence of gaits to be executed that lead to the goal. In these experiments we used an optimal Breadth First Search planner to extract the best known sequence of gait executions to the goal given the current observation and the desired observation. This planner generated the next observation by considering the outcome of each gait given the learned prediction rules whose conditions matched the current observation. The process was repeated until the desired observation was reached.

When a goal is directly observable the goal is defined in terms of the set of sensors given to the learner. Hence, a directly observable goal change can easily be handled by re-planning with the desired observation. On the other hand a hidden goal can only be sensed via a sensor that becomes true when the goal is reached, yet this sensor does not specify what the desired observation should be in terms of the sensors given to the learner.

Therefore, when the goal is hidden a goal management strategy must be used to maintain the observations associated with each goal (*goal observation*) in a dynamic list. Each time a goal is discovered, a new goal observation may be added to the list provided that a similar observation is not currently in the list. Two statistics corresponding to the number of successes and failures for each observation are recorded. The number of successes is incremented each time a similar goal observation is encountered, while the number of failures is incremented when the recorded observation is encountered but the goal is not found. The ratio of successes to failures for a goal observation indicates the probability of finding a hidden goal with it. Thus, the planner should test each goal observation in the list starting from the highest probability and explore the environment until it reaches the hidden goal, or runs out of tests and switches to random gait selection.

B. Environment's Configuration Changes

Through rule creation and maintenance SBL is able to learn new and useful knowledge regarding the environment. However, a mechanism is needed for forgetting obsolete or incorrect knowledge as the entities in the environment can easily change. This is especially crucial as SBL simultaneously explores all possible causes for a surprise, whereas many of them may subsequently prove to be incorrect. Rule forgetting is a technique derived to deal with such situations by rejecting obsolete or incorrect prediction rules.

Rules that are deemed inappropriate are marked as rejected such that they will not be used in future predictions or maintained any further. Nevertheless, they are kept in memory to serve as a reminder to avoid recreating them. When each new rule is constructed SBL checks it against all the valid rules as well as the rejected rules to ensure uniqueness prior to adding it to the prediction model. A pair of complementary rules may be deemed inappropriate if they cause contradictions or cause surprises consecutively and consistently.

A contradiction may arise immediately after rule splitting in situations where neither complementary rule describes the surprised result. This can be caused by two reasons: either the rule was created with a wrong c_0 in the first place (i.e. the entity or attribute c_0 was irrelevant to the current gait), or split with an incorrect prediction p_x. Either way, these two rules are self-contradicting and will not be pursued further in learning.

Each time a surprise occurs the complementary rules are refined to produce accurate predictions. However, if these rules fail consistently, then it can be inferred that the captured entity and attribute relations are inappropriate for the given context, meaning that the rules should be rejected. The ratio between the number of times a rule has been successful and the number of times it has been predicted can be stored against each rule as its probability of success. For example when this probability drops below 0.5 it means that the rule is failing more than 50% of the time. Therefore, a fixed cutoff probability such as 0.3 could be employed to effectively reject inappropriate rules. Rule rejection is performed during surprise analysis.

C. Gait & Sensor Changes

The first challenge presented by unexpected gait and sensor changes is that without the appropriate gait or sensor models it is difficult to establish that something unexpected has happened. In SBL, unexpected changes to gaits, sensors and the environment's configuration are detected as surprises. Then, entity and attribute relevance can be used to systematically establish and recover from it. Entity and attribute relevance is responsible for coupling entities and attributes with the relevant gaits. This could be situation or time dependent. A situation dependent example would be the robot using center camera coordinate attributes rather than the size attribute to avoid ambiguity when turning. A time dependent example would be the robot realizing that previously relevant gaits and sensors have now become irrelevant and vice versa, such as when gait definitions are switched, or the camera has been rotated.

Our approach to entity & attribute relevance is as follows. The relevance of entities and attributes are recorded in a table, in which each row uniquely identifies a sensor, an entity of that sensor, an attribute of that entity, a gait and an indicator corresponding to its relevance during learning. The relevance flag is updated depending on the following four scenarios:

1) If an entity or attribute does not change for a gait, then despite remaining active it will not be used in rule creation, splitting or refinement, for that gait. E.g. a constant room temperature sensor is irrelevant to the robot's motion.

2) For a given gait, if all rules that contain this entity or attribute as its first condition c_0 have been rejected, then the corresponding entity or attribute is flagged as irrelevant to that gait. E.g. a random value sensor will eventually cause all rules that use it to fail and be rejected.

3) For a given gait, when many rules containing the same entity or attribute consecutively fail, then the corresponding entity or attribute will be flagged as irrelevant to that gait. E.g. the size attribute will be deemed as irrelevant to turning as it causes consecutive failures when the proximity to objects vary.

4) When none of the active entities and attributes for a given gait register any change, then for future evaluation reactivate those flagged entities and attributes that indicated some change. A reactivated entity or attribute will be considered from the next learning cycle onwards. We call this reactivation as "forced relevance" which represents enlarging the attention of the robot. E.g. when the camera is unexpectedly rotated by 180º, the initial confusion will flag many entities and attributes as irrelevant. At some point the robot recognizes the lack of progress and re-evaluates its sensors (entities, attributes) and gaits in an attempt to improve its prediction model.

Entity & attribute relevance works during rule creation, rule splitting and refinement. SBL performs a lookup on the table and ignores any entities or attributes that are considered irrelevant during the analysis of surprises. When a rule is newly created, a copy of it (called trace rule) will be recorded to ensure that after the original rule is split, refined and rejected, it would not be recreated to repeat the same learning process. It is important to note that when forced relevance is invoked, SBL will purge the trace rules and rejected rules that refer to the toggled entities and attributes to facilitate relearning.

VI. Experimental Results

A. Experimental Setup

For testing purposes a SuperBot [14] modular robot was tasked at navigating in a typical office room as shown in Fig 3. A wheel attachment was used on the back dock of the module, while the front dock was connected to a passive frame constructed with pipes. SBL was executed on a laptop computer, which wirelessly communicated with SuperBot as the processing power of a single module is currently insufficient to cope with data from the camera.

Fig. 3. A SuperBot module attached to a passive frame

The set of gaits = {L, R} resulted in the robot turning left and right respectively over a small period of time. The camera sensor returned a dynamic set of entities corresponding to unique objects detected via a SURF [15] object detector. Each entity had 3 attributes {S, X, Y} corresponding to its size, x-location and y-location of the center of the object in camera coordinates. Goals were assigned by showing a desired observation to the robot, from which it extracted the objects of interest and their corresponding attributes. The set of operators {%, ~, ↑, ↓, <, <=, =, !=, >=, >} were defined.

B. Experimental Results

The robot was tasked with reaching a specified goal observation, which happened to be a particular book lying on the floor against a wall. A run was complete when the goal was reached, then a new run would commence by assigning another desired observation, which was another book placed at a different location. On subsequent runs these two goals were reassigned alternately. Due to limitations of battery power the number of runs was restricted to 8 and the averages of 3 separate tests are presented in Fig 4 & 5.

Initially, the robot was started with standard gaits and sensors with the two goal observations on either side. In run 1 SBL randomly executed gaits (as there is no model to start with) until the desired goal was visible as seen in Fig 4. In Fig 5 the high number of surprises for run 1 as opposed to run 2 indicates that the prediction model continued to be updated even while locating the second goal. In run 3 and run 4 the robot reached the now known goals after executing the appropriate gaits 12 times on average. Notice the lower number of surprises here indicates that the prediction model has become more accurate and no unexpected changes have occurred.

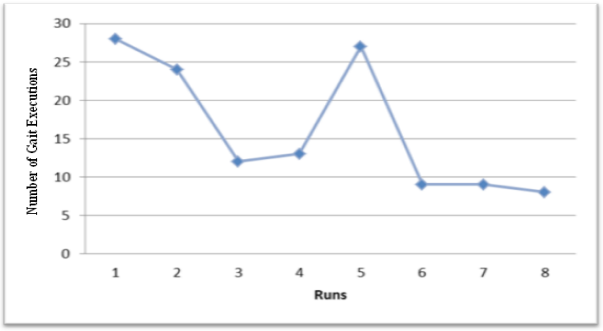

Fig. 4. Average change in number of gait executions

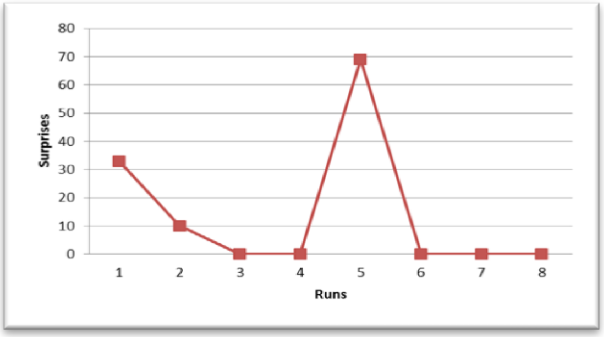

Fig. 5. Average change in surprises

At the beginning of run 5 several unexpected changes are introduced simultaneously. The gait definitions for L and R were swapped simulating a flip over, the camera was rotated by 90° and displaced by about 15cm along the top of the laptop, and one of the books in the goal observation was moved to another wall effectively bringing the two goals closer. The number of gait executions increased together with the number of surprises as SBL detected the unexpected changes. From run 6 onwards the model was sufficiently up-to-date so the robot executed the appropriate gaits 9 times on average to alternate between the goals. A video of a smaller test is available at http://www.isi.edu/robots/SBL/movies/unexpected.wmv.

The results from this real-world robot experiment confirms that SBL is reacting and adapting to simultaneous unexpected changes with real gaits, sensors and goals, in a natural environmental configuration in an unsupervised manner.

VII. Discussion & Future Work

This paper presented Surprise-Based Learning as a promising solution for reacting and adapting to simultaneous unexpected changes in a modular robot's gaits, sensors, goals and the environment's configuration in an unsupervised manner. SBL performs simultaneous representation and learning, by creating and maintaining a prediction model of the environment based on its interactions with the environment and its ability to detect and react to surprises. An important aspect of SBL is that it learns to discretize continuous sensors using comparison operators, so as opposed to some traditional learning techniques SBL is able to scale and adapt to changes in the state space as well as the approximation function autonomously. Comparisons will appear in future publications.

Modular robots have many configurations that will inevitably result in a varying number of gaits and sensors. First, it is not possible to enumerate all possible combinations of these and develop precise models that describe the outcome of each gait in a particular configuration in terms of a particular set of sensors in a given environment. Thus, learning is a necessary component for the ultimate solution. SBL can address this problem by autonomously learning a prediction model by executing the gait. Second, when an unexpected change occurs such as a sensor module being rotated or the entire robot flipping over SBL is able to detect changes through surprises and adapt its model without any user intervention.

As an example consider the butterfly gait where SBL would learn that executing it results in objects seen by the forward-facing camera getting larger while objects in the rear-facing camera get smaller, meaning that the robot is moving forward. So when the robot is tasked to reach an object directly in front, it will simply execute the butterfly gait. However, if the robot flips over, the same sequence of actuator commands would result in the robot moving backward, meaning that objects in the forward-facing camera become smaller while the rear-facing camera objects become bigger. This would cause many surprises in SBL, but the model will eventually be updated to reflect the new behavior. Using the updated model SBL would still be able to reach the object directly in front of it by executing some turn gaits followed by the butterfly gait.

In future, we plan to test several such scenarios to validate this claim further. Steps will be taken to distribute SBL across the modules to eliminate the need for external processing. We will expand the scope of "unexpected changes" to include module failures, change in the robot's configuration, and other important features of modular and self-reconfigurable robots.

To make this technique more robust, we will develop a graceful way to deal with *temporary interference* which may be a disturbance in a sensor's value that occurs infrequently over a short period of time, typically caused by sources outside of the robot's control, e.g. sunlight falling on an infrared sensor. Similarly, temporary interference of a gait is an infrequent short-term disturbance affecting the outcome of an actuator, e.g. slippage of a wheel. At present any temporary interference would result in a change of the model, whereas if the effects are temporary they should be ignored until they are established as permanent. We plan to achieve this by maintaining a probability to decide when model change is indeed necessary.

Acknowledgment

We would like to thank the reviewers & USC PRL group.

References

[1] A. Saffiotti, "Handling Uncertainty in Control of Autonomous Robots", Lecture Notes in Computer Science, pp. 198–224, 1997.

[2] M. Visinsky, "Fault Detection and Fault Tolerance Methods for Robotics", Rice University Thesis, 1991.

[3] C. Ferrell, "Failure recognition and fault tolerance of an autonomous robot", Journal of Adaptive Behavior, pp. 375-398, 1994.

[4] E. Kececi, X. Tang and G. Tao, "Adaptive actuator failure compensation for redundant manipulators", Robotica Journal, vol. 27, pp. 19-28, 2009.

[5] R. Murphy and D. Hershberger, "Classifying and Recovering from Sensing Failures in Autonomous Mobile Robots", In Proc. of the 13th National Conference on Artificial Intelligence, pp. 922-929, 1996.

[6] D. Stronger and P. Stone, "Towards Autonomous Sensor and Actuator Model Induction on a Mobile Robot", Journal of Connection Science, vol. 18, pp. 97–119, 2006.

[7] K. Doya, K. Samejima, K. Katagiri and M. Kawato, "Multiple Model-based Reinforcement Learning", Journal of Neural Computation, pp. 1347-1369, 2002.

[8] D. Pierce, B. Kuipers, "Map Learning with Uninterpreted Sensors and Effectors", Journal of Artificial Intelligence, vol. 92, pp. 169-229, 1997.

[9] M. Soika, "A sensor failure detection framework for autonomous mobile robots", In Proc. of the international conference on intelligent robots & systems, vol. 3, pp. 1735-1740, 1997.

[10] J. Bongard, V. Zykov and H. Lipson, "Resilient machines through continuous self-modeling", Science, vol. 314, 1118-1121, 2006.

[11] E. Stone, M. Skubic, J. Keller, "Adaptive Temporal Difference Learning of Spatial Memory in the Water Maze Task", In the proc. of the International Conference on Development and Learning, pp.85-90, 2008.

[12] N. Ranasinghe, W-M. Shen, "Surprise-Based Learning and experimental results on robots", International conference on developmental and learning, June 2009.

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

[14] B. Salemi, M. Moll and W.-M. Shen, "SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System", Intelligent Robots and Systems, pp. 43-52, 2006.

[15] H. Bay, T. Tuytelaars and L.-V. Gool, "Surf: Speeded up robust features", European conference on computer vision, pp. 404-417, 2006.