2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Curiosity-Based Learning Algorithm for Distributed Interactive Sculptural Systems

Matthew T. K. Chan, Rob Gorbet, Philip Beesley and Dana Kulić

Abstract — **The ability to engage human observers is a key requirement for both social robots and the arts. In this paper, we propose an approach for adapting the Intelligent Adaptive Curiosity learning algorithm to distributed interactive sculptural systems. This Curiosity-Based Learning Algorithm (CBLA) allows the system to learn about its own mechanisms and its surroundings through self-experimentation and interaction. A novel formulation using multiple agents as learning subsets of the system that communicate through shared input variables enables us to scale to a much larger system with diverse types of sensors and actuators. Experiments on a prototype interactive sculpture demonstrate the exploratory patterns of the CBLA and collective learning behaviours through the integration of multiple learning agents.**

Keywords— robots and embodied art, reinforcement learning, intrinsic motivation

I. INTRODUCTION

Interactive arts are a type of art form that requires the involvement of the spectators to achieve its purpose. With recent advances in capabilities and miniaturization of computers, sensors and actuators, artists now have access to more tools to create highly complex interactive artworks. In the Hylozoic Series of kinetic sculptures built by Philip Beesley Architect Inc., the designers use a network of microcontrollers to control and sample a sizable number of actuators and sensors [1] [2]. Each node in the network can perform a simple set of interactive behaviours. While the behaviours have previously been either pre-scripted or random, complex group behaviours have been seen to emerge through the communication among nodes and interaction with the spectators [3].

In this paper, we introduce the Curiosity-Based Learning Algorithm (CBLA) to replace pre-scripted responses in the Hylozoic series installation. The CBLA re-casts the interactive sculpture as a set of agents driven by an intrinsic desire to learn. Presented with a set of input and output variables that it can observe and control, each agent tries to understand its own mechanisms, its surrounding environment, and the occupants, by learning models relating its inputs and outputs. We hypothesize that the occupants will find the behaviours which emerge during CBLA-based control to be interesting, more life-like, and less robotic.

This approach reduces the reliance on humans to manually design interesting and "life-like" behaviours. In systems with large numbers of sensors and actuators, programming a complex set of carefully choreographed behaviours is complicated and requires lengthy implementation and testing processes. Furthermore, this approach allows the sculpture to adapt and change its behaviour over time. This is especially interesting for permanent installations in which the same users may interact with the sculpture over an extended period of time.

This work was supported by the Natural Sciences and Engineering Research Council.

All authors are with the University of Waterloo, Waterloo, ON, Canada (email: matthew.chan@uwaterloo.ca).

A. Related Work

Interactive arts can be categorized as Static, Dynamic-Passive, Dynamic-Interactive, and Dynamic-Interactive (varying) based on the degree of the interaction between the art works and the viewers [4]. Dynamic-Interactive systems give the human viewers an active role in defining the behaviours of the system. This category introduces an agent that modifies the specifications of the art object. This additional unpredictability introduces a new dimension of complexity to the behaviours of the system.

In [5], Drummond examined the conceptual elements of interactive musical arts. For an interactive system to move beyond being simply a complex musical instrument with a direct reactive mapping from inputs to generation of sound, it must possess some level of autonomy in the composition and generation of music. In addition, the interactivity should be bilateral; the performer influences the music and the music influences the performer. These concepts can easily be extended to visual and kinetic arts. Visual-based interactive systems such as the Iamascope in Beta_space [6] and audio-based systems such as Variations [7] engaged the participants by allowing them to directly affect the output of the system. Works in interactive architecture [1] [8] try to provide a responsive and immersive environment where the viewers can feel as if they belong to the system.

However, most of these works are the non-varying type of Dynamic-Interactive system, as their responsive behaviours do not change. Over a longer term, the system will become more predictable and its effectiveness in engaging the users will consequently decrease. In this work, we aim to create a varying interactive artwork by emulating the characteristics of living organisms such as curiosity and learning [3].

To emulate life-like behaviours, one can start by observing how human beings behave. [9], [10] modelled how human beings convey or mask their intentions through movement and applied these models on a humanoid robot. Similarly, [11] focuses on making the robot's motion more understandable by emulating the coordinated effects of human joints. Those studies focus their attention on making the intent of the robots clear. In contrast, our objective is to make robots more engaging and life-like, where unpredictability might be a desirable quality. For instance, [10] showed that the robot's perceived intelligence increased when the participants believed that the robot was intentionally deceptive. Our work investigates whether unpredictable

behaviours emerging from the learning process will appear more life-like and engaging.

One of the open questions in artificial life research is whether we can demonstrate the emergence of intelligence and mind [12], examined in projects such as the Petting Zoo by Minimaforms [13] and Mind Time Machine [14]. The idea of emergence of structure and consciousness is explored in many previous works in the field of developmental robotics [15] [16] [17]. Oudeyer et al. developed a learning mechanism called Intelligent Adaptive Curiosity (IAC) [18], a reinforcement learning algorithm with the objective of maximizing learning progress. In his experiments, he showed that an agent would tend to explore state-space that is neither too predictable nor too random, mimicking the intrinsic human drive of curiosity, which continually tries to explore areas that have the highest potential for learning new knowledge. However, previous work did not cover how the IAC might be scaled to a distributed system with a large sensorimotor space.

This paper builds on Oudeyer's IAC and applies it on an architectural-scale distributed interactive sculptural system. This requires a control architecture that can communicate and synchronize with many individual nodes.

By having multiple learning agents that focus on different subsets of the system, the dimensionality of each agent's sensorimotor space can be reduced. Furthermore, multiple learning agents can be set up to run in parallel but at different frequencies in order to accommodate different actuator bandwidths. These different agents are coupled through shared sensory channels and their operating environments, leading to collective learning through individual explorations.

II. Prototype Interactive Sculptural System

The CBLA requires a large increase in sensing and control capability which previous versions of the Hylozoic Series embedded electronics [1] are unable to provide. A new Hylozoic series was developed with re-designed hardware, to enable the control and sampling of a larger number of actuators and sensors at a higher frequency.

A. Physical Hardware

The sculpture system is assembled from a number of clusters. Physically, each cluster comprises 3 Tentacle devices and 1 Protocell device. A *Tentacle device* is composed of two shape memory alloy (SMA) wires and an LED as actuators, and two infrared (IR) proximity sensors and a three-axis accelerometer as sensors. When actuated, the two SMA wires pull a lever, which mobilizes a frond-like arm. The accelerometer is mounted on the arm and serves as the proprioceptive sensor for the SMA wires. A *Protocell device* comprises a high-power LED as actuator, and an ambient light sensor for proprioception. Figure 1 shows how they are mounted spatially. The sensors are both proprioceptive, sensing the actuated movement, and environmental, sensing the external environment. For instance, an accelerometer attached to a Tentacle Arm senses both the movements caused by activations of the SMA wires and by human contact.

A. Control Architecture

The control software of the sculpture consists of a low-level layer and high-level layer, connected physically through USB. The low-level layer consists of the firmware that interfaces with the peripherals that connect with the actuators and sensors, which resides on a set of Teensy 3.1 USB-based development boards (www.pjrc.com). The high-level layer consists of the tools that facilitate communication with the low-level layer, and the application that dictates the system behaviour. The abstraction provided by the high-level layer allows flexibility in defining the nodes and their relationship to each other.

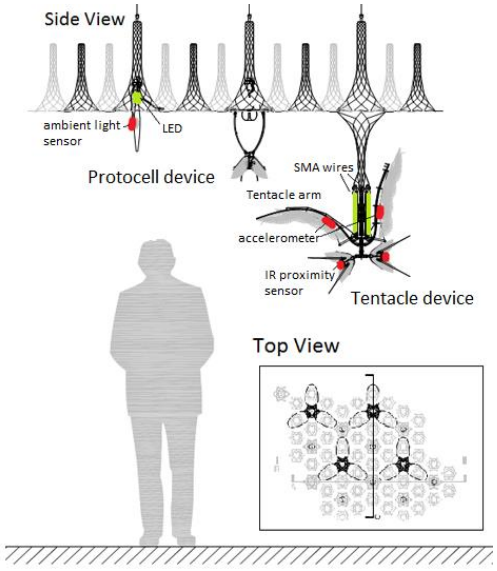

Figure 1. Spatial configuration of the physical components (figure adapted from an image provided by Philip Beesley Architect Inc.).

Figure 2 illustrates the relationship between the different software components. The high-level and low-level layers share a set of output parameters that govern the behaviours, and input parameters that represent the states of the system.

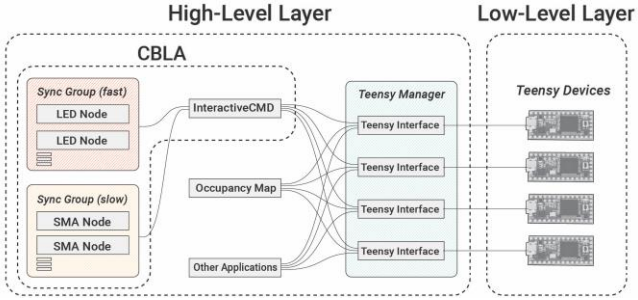

Figure 2 The low-level layer consists of the Teensy devices which interface with the physical hardware (not shown in the figure). On the high-level layer, a Teensy Interface thread is created for each Teensy, to communicate with the applications. The Teensy Manager may also be called to perform system-wide functions. The CBLA is one application that can control the sculptural system. Other applications may also be run in parallel (e.g., an Occupancy Map generator, shown but not discussed here).

The cluster system is abstracted into two types of nodes: Tentacle Node and Protocell Node. Each node is characterized by a configuration and a set of input and output variables. Specifications of the nodes are presented in the experiment sections.

III. Curiosity-based Learning Algorithm

In this section, we first briefly describe the Intrinsic Adaptive Curiosity (IAC) algorithm proposed by Oudeyer [18]. Then, our approach for adapting and generalizing the algorithm for implementation on a distributed kinetic sculpture is presented. This includes additional parameters, the introduction of an idle mode, and the integration of multiple learning agents in a distributed network.

A. Intrinsic Adaptive Curiosity

In [18], Oudeyer's goal was to develop a robot that is capable of life-long learning. The robot makes sense of the mapping between its actuators and sensors through continuous self-experimentation, without explicit instruction from human experts. Fundamentally, the IAC is a reinforcement learning algorithm with an objective to maximize learning progress. Learning progress is defined as the reduction in prediction error. In other words, the agent seeks to explore the region of the sensorimotor space that leads to the highest improvement in prediction error. As a result, the algorithm avoids learning in the parts of sensorimotor space that are either too random or too predictable. This formulation leads to continual change in behaviour over time, as regions of the state space that are initially interesting become less interesting as the system becomes more knowledgeable about them.

The system consists of two learning mechanisms, the Classic Machine learner (classM) and the Meta Machine learner (metaM). Based on the context (the sensor inputs) and the action (the actuator outputs), classM computes a prediction of the consequence, i.e., the sensor values at the next time step. Then, it compares the actual consequence with the prediction and modifies its model to reduce the error in the subsequent prediction. metaM predicts the error of classM, i.e., how accurately classM is able to predict the consequence. The actual prediction error is then fed back to metaM, which modifies its estimate of prediction error of the newly modified classM. This change in the prediction error is recorded at each time step in the Knowledge Gain Assessor (KGA). The KGA then computes the expected learning progress by calculating the change in error estimation. This expected learning progress is used as the reward.

In Oudeyer's paper [18], one important feature is the idea of regional experts. Each region collects exemplars of similar sensorimotor context, and has an expert that is trained on the exemplars in the region. Exemplars are the training data for the prediction model and they are collected as the system selects actions and observes the consequences. The "features" are the sensory inputs *S(t)*, and selected actions *M(t)* at time *t*; and the "labels" are the resultant sensory inputs *S(t+1)* at time *t+1*. The regional experts constrain the estimate of learning progress within their respective sensorimotor contexts. This is important because it allows each expert to use a simpler model, as it covers only a small region of the state space.

The following are the steps for the learning algorithm:

1. Read sensory inputs *S(t)*
2. Select action *M(t)* with the highest predicted learning progress *R(SM(t))* based on ε-greedy selection policy
3. Consult expert specialized in the relevant sensorimotor context *SM(t)* to predict the expected sensory inputs *S(t+1)'*
4. Perform action *M(t)*
5. Observe actual sensory inputs *S(t+1)* and add to the expert's training set
6. Compute prediction error *e(t+1) = S(t+1) – S(t+1)'*
7. Compute the change in prediction error *R(SM(t)) = -[e(t+1) - e(t)]*
8. Update the learning progress for the sensorimotor context *R(SM(t))*
9. Repeat 1 to 8 indefinitely

B. CBLA Engine

Oudeyer [18] observed during IAC experiments that the resulting robot behaviour had similarities to children's learning and playing behaviours. We hypothesize that this type of learning mechanism is well suited to interactive architecture systems, as the learning process itself will generate interesting behaviours and be an interesting feature of the installation to the visitors. We now adapt the IAC algorithm [18] to implement a learning architecture on the distributed architectural system described in Section II. Each system node runs its own version of the algorithm, called the CBLA engine. It is comprised of a node, an action selector, and an expert, as illustrated in Figure 3.

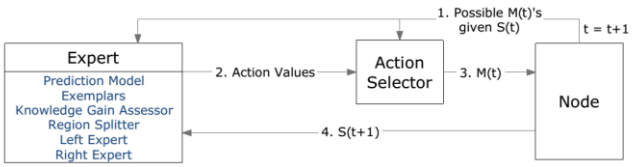

Figure 3. (1) At the start of each time step, the Node outputs the current possible actions *M(t)*'s given *S(t)*. (2) Then, the Expert provides the Action Values associated with those possible *M(t)* to the Action Selector. (3) After that, the Action Selector selects an action and actuates the Node. (4) After the Node has performed the action, it returns the actual resultant state *S(t+1)* to the Expert. The Expert can then improve its internal prediction model and update the Action Value associated with *SM(t)*.

Node: A node represents a subset of the sculptural system. Each node is specified by set of sensor input and actuator output variables, and its own set of configuration parameters and functions, embodying the characteristics of the physical system that it represents. It communicates with the lower-level layer to set the actuator outputs and sample the relevant sensory inputs. The type and number of sensory inputs and actions are configurable, example configurations will be described in more detail in Section IV.

Action Selector: The action selector selects an action based on a list of possible actions, *M(t)*, and their associated action values. It either selects an action that has the highest value (exploitation), or selects an action from the list at random (exploration). In the experiments presented in this paper, a version of the adaptive ε-greedy [19] action selection method was used. ε specifies the probability that a random action is selected instead of the action with highest action value. The magnitude of ε changes based on the magnitude of the action value. When the action value is low, the probability of selecting a random action is high since the current expected

reward is low and it's worth spending time exploring new regions that may lead to higher rewards.

Expert: An Expert consists of a prediction model, a set of exemplars, a Knowledge Gain Assessor (KGA), a region splitter, and potentially two child experts. The prediction model models the relationship between the node's input and output variables. In all the experiments presented in this paper, linear regression was used. At every time step, the model is used to make a prediction about the immediate future state based on the current state and the selected action. This prediction model is trained on the set of exemplars that were previously observed. These exemplars are represented as *[SM(t),S(t+1)]* pairs. *SM(t)* represents the sensorimotor context, i.e., the concatenation of the sensory state and the action taken at time *t*. *S(t+1)* represents the observed sensory state at time *t+1*. The KGA calculates the reward given the predicted and actual sensory inputs.

Figure 4 illustrates the internal workings of the KGA. The *metaM* returns the average error over a window of time τ time steps ago, $<e(t+1)\text{-}\tau>$. The reward is calculated by subtracting the current mean error, $<e(t+1)>$. The action selector then computes the action values using this reward.

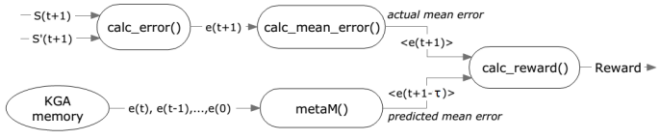

Figure 4. *S(t+1)* and *S'(t+1)* are the actual and predicted sensor input variables. (1) The KGA computes the error as the root-mean-square of their difference. (2) After that, it computes the mean error over a window of previously calculated errors. Note that the mean error is only calculated based on errors associated with this particular region. (3) The *metaM* predicts the error by taking the mean error over time. (4) Finally, the KGA outputs the Reward by subtracting the actual mean error from the predicted mean error.

An expert represents a region in the sensorimotor space. Each prediction is generated by the expert in the region with the most similar sensorimotor context. Figure 5 shows the internal mechanism of an Expert.

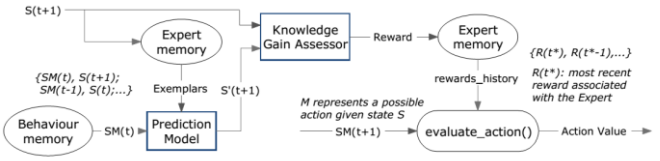

Figure 5. (1) The prediction model of the Expert first makes a prediction of the resultant sensor input *S(t+1)'* based on the current sensorimotor context, *SM(t)*. (2) This prediction and the actual sensor input *S(t+1)* is then fed into the KGA. *S(t+1)* is also added to the collection of exemplars, which the prediction model is trained on at every time step. (3) After that, the KGA computes the reward and updates the Action Value associated with *SM(t)* in the Expert memory. These Action Values are recalled to evaluate a possible action in the next time step the Expert is active.

The system initially has only one region. All new exemplars are added to the same region. As more exemplars are added to memory, a region will split into two parts if it contains a sufficient number of exemplars and if prediction error is high. This prevents learned models with high accuracy from being split indefinitely.

If the split criteria are met, the Region Splitter finds a cut value and a cut dimension that minimizes the average variance in each sub-region. In the current implementation, 100 random cut values are generated for each dimension and the corresponding variances of the resultant sensor inputs are computed. The cut value and dimension pair with the lowest variance is selected, and the exemplars of the parent region are split into the two regions based on the cut dimensions and values. The high-level mechanism of the region splitting process is illustrated in Figure 6.

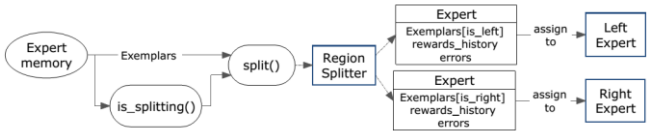

Figure 6. When the function "split()" is called, it first checks if the split criteria are met by calling "is_splitting()". If the split criteria are met, it forwards the exemplars to the Region Splitter. The Region Splitter then splits the exemplars into two and assigns them to the Left and Right Expert. Other properties such as the previous errors and rewards are passed on as well.

The overall process is described in the pseudocode for the CBLA and is shown in Figure 7.

```
t = 0
S(t) = S(0)
Loop:
   [Possible_M(t)] = Node → get_possible_action(S(t))
   [Action_Value(M(t))] = Expert → evaluate_action(Possible_M(t))
   M(t) = Action_Selector → select_action([Possible_M(t), Action_Value])
   Node → actuate(M(t))
   S(t+1) = Node → report()
   S'(t+1) = Expert → predict(S(t), M(t))
   Expert → append(S(t), M(t), S(t+1), S'(t+1))
   Expert → split()
   t = t + 1
```

Figure 7. Pseudocode for the CBLA

1) Differences between the CBLA Engine and the IAC

In developing the CBLA Engine described above, several adaptations have been made to the IAC algorithm to enable application to the kinetic sculpture installation.

First, several improvements are made to the logic of the Region Splitter. As we run the algorithm over an extended period of time, the expert might not be able to improve the prediction model any further given the available sensory information. Since Oudeyer's method simply splits a region when the number of exemplars exceeds a certain threshold, it will continue to split regions even when the expert associated with that region has already acquired a good model with low prediction error. This is undesirable as it leads to over-fitting and makes the system harder to adapt when the system itself or the environment changes. In our implementation, an additional prediction error threshold is added to prevent regions from splitting when the prediction error is low. Moreover, if the split does not improve the prediction accuracy, there is no reason to split the region. After the split, the split quality measured by the magnitude of the average within-group variance must also be above a threshold. If it is not, the split will be retracted. During the early stages of the learning process, learning opportunities are plenty. Setting the split number threshold too high can hamper the speed of learning. However, over time, easy learning opportunities diminish and complex regions require a larger number of exemplars. Thus, the split number threshold grows at a

constant rate every time a split happens. This way, less explored regions can maintain low thresholds that promote learning, while mature regions have higher thresholds that promote gathering more exemplars and reduce over-fitting.

Second, visually, it is difficult to distinguish between the periods when the CBLA is learning from the periods when the CBLA is simply executing random actions after all regions are learned. In order to better demonstrate the learning process, an idle mode is introduced. In our experiments, the idle mode action is chosen as the action that requires the least power. This gives the visitors the sense that the sculpture is resting. During idle mode, the system selects the idle action a majority of the time. Otherwise, it will select an action based on its regular action selection policy. The CBLA engine enters idle mode when the average highest reward over a window of time is lower than a threshold. Conversely, it exits idle mode when the average highest reward, or the change of it, over a window of time is higher than a threshold. Once it exits idle mode, it must stay in non-idle mode for at least a certain period of time before being allowed to enter idle mode again.

C. CBLA System

One option for controlling a larger system of sensors and actuators is to control them centrally through a single CBLA engine. Yet, if all output and input variables are grouped into one node, the sensorimotor state will become very large and it may take a long time for the system to converge and react to changes. Thus, subsets of related variables are grouped into nodes. Each node runs on its own CBLA engine in parallel.

There are several options for grouping sensors and actuators into nodes. One approach is to group actuators and their associated proprioceptive sensors by function, because these relationships are the easiest to model. In addition, if multiple actuators work together to perform one function that directly affects one sensor, they should be grouped as a node because their relationship must be considered in order to produce a prediction model for the sensor input. However, grouping simply by function cannot capture environmental changes and occupant interaction effectively.

Another approach is to group sensor input and actuator output variables by spatial proximity. Since environmental changes and occupant interaction are likely to affect components that are close together physically, this will allow the system to capture those dynamics more effectively.

However, all sensor input and actuator output variables associated with a particular node are updated once per loop. This means that the loop speed is limited by the slowest actuator or sensor within the node. For instance, an SMA-actuated Tentacle node has heating and cooling time of about 12 seconds, while an LED can be turned on and off many degrees of magnitude faster. This is very inefficient and limiting for the fast components. Therefore, components running at different speeds should be grouped into different nodes. This allows each node to run at its own speed, independently from the other nodes.

In our implementation, each node is constructed of components that are related functionally and proximally. For instance, LED output and ambient light sensor are in one node; Tentacle motion and accelerometer are in one node. On top of that, nodes that can be run at similar speed are grouped into one synchronization group. Different synchronization groups run at different frequencies and capture the dynamics of the system at different time scales.

Nevertheless, if the nodes perform learning independently, the CBLA does not actually capture the dynamics of the entire system. To integrate the nodes into one coherent entity, we have used shared sensors.

By using input variables relating to a common sensor, system changes effected by one node can be detected by another node indirectly. In our system, the Protocell node and Tentacle nodes share an IR proximity sensor. Figure 8 illustrates the connections and groupings used in this experimental test bed.

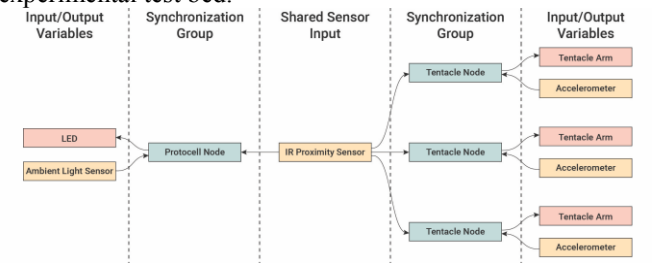

Figure 8. Nodes' connections and groupings

IV. Experiment 1: Demonstration of Exploratory Pattern of the CBLA on a Protocell Node

Although the CBLA was designed for a multi-node distributed system, it can be difficult to visualize the modeling process due to the high-dimensionality of the data and the models. To demonstrate the action selection pattern and the learning process, it was first tested on a simple toy example that can easily be visualized in 2-dimensional space. In this experiment, idle mode was disabled as the main objective was to observe and verify the exploration pattern of the CBLA.

The system in this experiment consists of a Protocell node; a single-input, single-output system. For this system, S is a scalar measurement from an ambient light sensor, recorded directly as a 12-bit value. M corresponds to the voltage duty cycle supplied to the LED, with 0 being completely off (0V) and 100 being the maximum allowable voltage (~4.7V). The loop period was set to 0.05s. In this experiment, the system ran for 2500 time steps without any external interference.

Based on the reward structure, which favours learning first the most predictable regions of the state-space, the CBLA should first explore the regions of the sensorimotor space that have low variance. Once the model in that region is learned, it should move on to areas with higher variance.

Figure 9 shows the evolution of the prediction model and actual exemplars over time. As expected, the CBLA first selected actions associated with lower LED output levels, which were within the low variance regions. Once the models in the low brightness regions were acquired, it moved toward higher brightness regions. Figure 10 shows that the best action and the actual selected action were completely random at first ($T<400$). The system then focused on the easy-to-learn, lower brightness level. After that, it moved toward the hard-to-learn, higher brightness regions when it saw little improvement in the low-brightness regions ($T=1500$). After a period of

exploration in the high-brightness regions, their associated prediction errors were reduced, and the system returned to the low-brightness regions (T=1900). The resulting pattern of activation was interesting visually, as it resulted in non-random activations that have the potential to convey a notion of intent to the viewers.

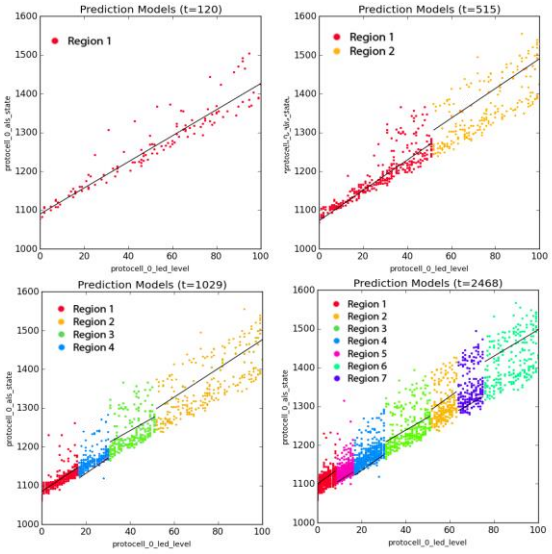

Figure 9. Evolution of the prediction models. Each point represents an exemplar. Points with the same colour are in the same region and the black lines are the cross-section of the linear models at S(t) = 0. The regions are numbered in the order that they were created.

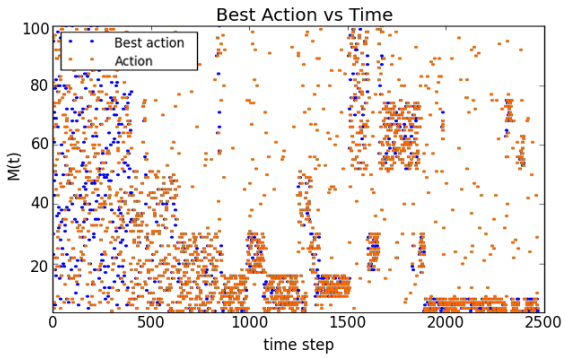

Figure 10. Action vs Time Graph; the *y*-axis is the output of the LED *M(t)* and the *x*-axis is the time step. Orange dots represent the actual action taken and blue dots represent the best action. The best action is defined as the action with the highest action value given the sensorimotor context. Non-best actions are selected occasionally in order to explore the state space.

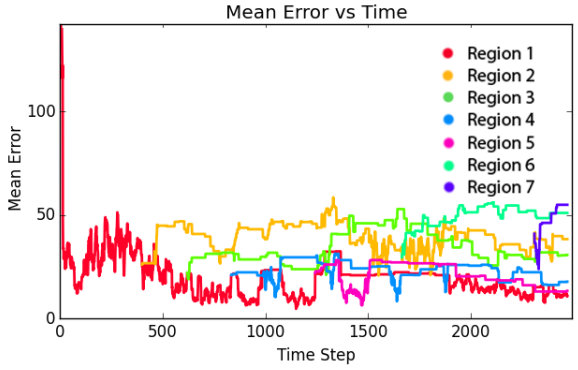

Figure 11. Mean error vs time graph. Each colour represents a region and the colour code corresponds to final prediction model graph in Figure 10.

Figure 11 shows the mean error vs. time graph. The prediction error quickly dropped to a relatively low level. To improve its prediction further, the state-space was split into regions with low and high error. This allowed Region 1 (low variance region) to further reduce its prediction error.

Figure 12 shows that the action value of a region does not stay constant. As the prediction improves, the value of actions in that region decreases over time as the region becomes "learned" and further learning potential decreases.

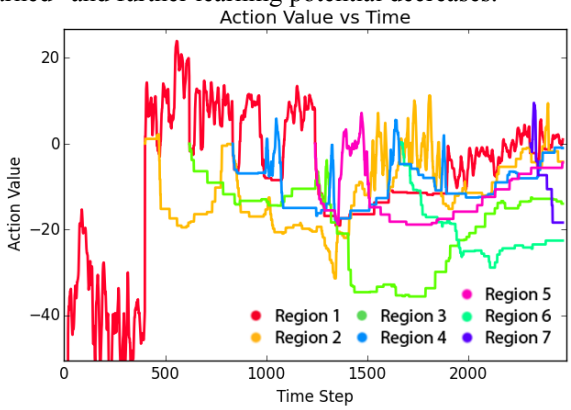

Figure 12. Action value vs time graph

V. Experiment 2: Tentacle and Protocell Nodes

In this Section, we describe a demonstration of an integrated system consisting of both light and SMA actuation mechanisms. In this experiment, there is one Protocell node and three Tentacle nodes. The Protocell node was the same as in Section IV, with the addition of an IR proximity sensor. For the Tentacle node, the input variables are the average accelerometer readings of the three axes, and the IR proximity sensor reading over the 12.5s loop period; the output variable is the action of the Tentacle Arm. There are four discrete actions: rest, lower to the right, lower to the left, and lower to the centre. Note that in this set up, the two types of nodes run with different loop periods, but coupling between them is accomplished through the shared IR proximity sensor, which measures occupant proximity as a 12-bit value.

The system runs undisturbed until, after some initial learning, all of the nodes enter idle mode. During this time, the IR proximity sensor is pointed toward an empty area. Then, a participant approaches the sculpture space and is detected by the IR proximity sensor. The system should then exit idle mode and begin learning a new model that captures the change in the environment. Since the IR sensor is shared by all nodes, they are all expected to recognize the change and exit idle mode at approximately the same time.

Figure 13 shows how the action values change over time for each node. The coloured lines represent the action values and each colour represents a region. The blue dots underneath the plot indicate when the node was idle.

At first, all the nodes started learning their own models, and then entered idle mode. At around 390s, a human participant walked in front of the IR proximity sensor. This triggered a large reduction in action value at first, due to an increase in prediction error. However, as more data was collected, the action values for all four nodes quickly jumped

up. This prompted the nodes to exit idle mode and begin generating actions in an attempt to learn the model. After a period of readjustment, all nodes re-entered idle mode after the new environment was learned. Videos of this experiment can be viewed in the accompanying video submission.

From the video and Figure 13, one can see that the Protocell node and the three Tentacle nodes reacted to the environmental change nearly simultaneously. They exited the idle state and shifted to more exploratory actuation patterns. This showed that the shared sensor input variable was able link the CBLA engines together, even though they run independently at different frequencies. This experiment demonstrates that the reaction of the system to the changed environmental conditions creates an interaction with the visitor without any explicitly pre-programmed behaviours. The system's intrinsic curiosity drives itself to perform actions and elicit responses from this new environment with the participant's presence, and update its prediction model. We anticipate that the visitors will find such behaviours engaging as the visitors can recognize that the sculpture is responding to their presence and action but they would not be able to easily predict how it might respond. This quality provides the CBLA System the potential to be more life-like than a pre-scripted or a random system.

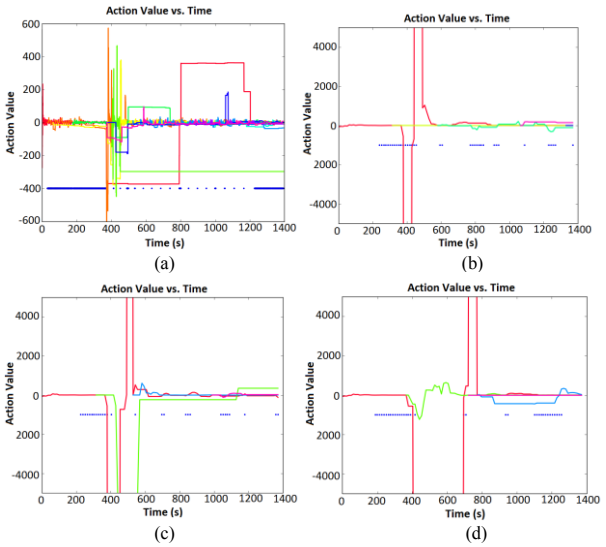

Figure 13. Action-value vs Time graph for the Protocell node (a) and the three Tentacle nodes (b), (c), (d).

VI. Conclusions and Future Work

Motivated by the desire to make interactive art sculpture more life-like and engaging in the long term, we adapted the concept of curiosity-based learning from developmental robotics for a distributed system with abstract forms and large numbers of sensors and actuators. By grouping subsets of input and output variables based on their functions, spatial proximities, and cycle times, each learning agent can mitigate the curse of dimensionality and operate with a configuration suitable for its component characteristics, while forming a coherent entity through shared sensors.

We presented two experiments demonstrating the exploration patterns of the CBLA and testing the multi-node CBLA system integrated by a shared sensor. These promising results demonstrated the possibility of applying intrinsic curiosity-based learning on large-scale distributed system.

In the future, we plan to conduct further experiments to examine and characterize the exploratory patterns and human perception of the emergent behaviours of a CBLA system with a larger number of nodes under various configurations.

References

[1] P. Beesley, Hylozoic Ground: Liminal Responsive Architectures, Toronto, ON, Canada: Riverside Architectual Press, 2010.

[2] P. Beesley, Kinetic Architectuers & Geotextile Installations, Toronto, ON: Riverside Architectural Press, 2007, 2010.

[3] P. Beesley, Sibyl, Toronto, ON: Riverside Architectural Press, 2012.

[4] E. Edmonds, G. Turner and L. Candy, "Approaches to interactive art systems," in *The 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, Singapore, 2004.

[5] J. Drummond, "Understanding Interactive Systems," *Organised Sound,* vol. 14, no. 02, pp. 124-133, 2009.

[6] B. Costello, L. Muller, S. Amitani and E. Edmonds, "Understanding the Experience of Interactive Art: lamascope in Beta_space," in *the second Australasian conference on Interactive entertainment* , Sydney, Australia, 2005.

[7] B. Wands, "Variations: an interactive musical sculpture," in *The 5th conference on Creativity & Cognition*, London, UK, 2005.

[8] Hangar.org, "Interactive Architecture," [Online]. Available: http://www.hangar.org/docs/docs_lab/IA.pdf.

[9] A. D. Dragan, S. Bauman, J. Forlizzi and S. S. Srinivasa, "Effects of Robot Motion on Human-Robot Collaboration," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2015.

[10] A. Dragan, R. Holladay and S. Srinivasa, "An Analysis of Deceptive Robot Motion," in *Robotics: Science and Systems*, 2014.

[11] M. J. Gielniak, K. C. Liu and L. A. Thomaz, "Generating human-like motion for robots," *The International Journal of Robotics Research,* vol. 32, no. 11, pp. 1275-1301, 2013.

[12] M. A. Bedau, J. S. McCaskill, N. H. Packard, S. Rasmussen, C. Adami, D. G. Green, T. Ikegami, K. Kaneko and T. S. Ray, "Open Problems in Artificial Life," *Artificial Life,* vol. 6, pp. 365-376, 2000.

[13] Minimaforms, "Petting Zoo FRAC Centre," Minimaforms, [Online]. Available: http://minimaforms.com/petting-zoo-frac/.

[14] T. Ikegami, "A Design for Living Technology: Experiments with the Mind Time Machine," *Artitificial Life,* vol. 19, pp. 387-400, 2013.

[15] M. Lungarella, G. Metta, R. Pfeifer and G. Sandini, "Developmental robotics: a survey," *Connection Science,* vol. 15, no. 4, pp. 151-190, 2003.

[16] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino and C. Yoshida, "Cognitive Developmental Robotics: A Survey," *IEEE Transactions on Autonomous Mental Development,* vol. 1, no. 1, pp. 12-34, 2009.

[17] V. R. Kompella, M. F. Stollenga, M. D. Luciw and J. Schmidhuber, "Explore to See, Learn to Perceive, Get the Actions for Free: SKILLABILITY," in *International Joint Conference on Neural Networks*, Beijing, China, 2014.

[18] P.-Y. Oudeyer, F. Kaplan and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evoluntionary Computation,* vol. 11, no. 2, pp. 265-286, 2007.

[19] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," *KI 2010: Advances in Artificial Intelligence,* vol. 6359, pp. 203-210, 2010.