Action Chaining by a Developmental Robot with a Value System

Yilu Zhang and Juyang Weng *

Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{zhangyil,weng}@cse.msu.edu

Abstract

A developmental cognitive learning architecture with a value system is proposed for an artificial agent to learn composite behaviors upon the acquisition of basic ones. This work is motivated by researches on classical conditioning in animal learning areas. Compared to former works, the environment is not known and the proposed architecture lets an agent conduct learning through online real-time experiences. All possible perceptions and actions, including even the actual number of classes, were not available until the programming was finished and the robot started to learn in the real world. Experiments with our SAIL robot are reported to show how a trainer instructed (or shaped) the behaviors of the agent through verbal commands.

1 Introduction

Animals are very competent in learning complex behaviors upon acquired basic behaviors, which, undoubtedly, is a highly desirable and challenging capability for a man-made robot. According to studies in animal learning, classical conditioning as well as other conditioning protocols plays an important role in the animals' behavior scaling-up process. These conditioning protocols enable animals to take advantage of the orderly sequence of events in the environment and learn new stimulus-response associations. Mathematically, classical conditioning can be described in the following so-called *acquisition procedure* (A.P.),

$$CS \rightarrow US \rightarrow UR \Rightarrow CS \rightarrow UR$$

where, CS stands for conditioned stimuli, US for unconditioned stimuli, UR for unconditioned response, $\rightarrow$ means "followed by", and $\Rightarrow$ means "develops". Other conditioning protocols that have been widely studied include secondary conditioning, where one CS is preceded by another CS,

$$CS_2 \rightarrow CS_1 \rightarrow UR \Rightarrow CS_2 \rightarrow UR$$

and instrumental conditioning, where changes in behavior occur as a result of the positive or negative consequences the behavior produces [2].

Machine learning researchers have been interested in these conditioning protocols for a long time. Many computational models have been studied (see [1] for a survey). While these works gave valuable insights into classical conditioning, they were typically validated in synthesized symbolic domains instead of a robot which took raw sensory signals in real-time. When realizing classical conditioning and other protocols on embodied artificial agents, to reduce the difficulty level, sensations or responses were usually predefined. For example, in [8] [5], a robot functioned upon a set of perception and behaviors which were defined symbolically in advance. In [6], a mobile robot learned to associate the visual sensations of "blob" and "strip" patterns with the unconditioned responses of taste sensations (simulated by conductivity). Three neural networks were designed specifically to recognize the predifined two types of "strip" patterns and one type of "blob" patterns. As far as we know, there has been little prior published attempt on the very challenging scenarios where the perceptual or action classes are not only unknown, their type and number are also open, as what animals experience in classical conditioning contexts.

In our previous works on the SAIL developmental robot project, we have demonstrated SAIL's vision-related and audition-related learning capabilities [10] and [11]. All the learning processes of SAIL was con-

*The authors would like to thank Wey S. Hwang for his major contribution to an earlier version of the IHDR program. The work is supported in part by the National Science Foundation under grant No. IIS 9815191, and DARPA ETO under contract No. DAAN02-98-C-4025, DARPA ITO under grant No. DABT63-99-1-0014.

ducted online in real-time through *physical interactions* between trainers and the SAIL robot. Without any task-specific information available, such as visual or acoustic models about the world or task, the SAIL developmental program generated internal representations and architecture autonomously while events occurred. This is a critical step for robots to learn new tasks without reprogramming. The importance of this line of work, as well as the great challenges, have been recognized lately [4] [3]. In [4], a multimodal learning system was built but the learning process was not done strictly through real-time interactions between a robot and the environment yet. The interesting work in [3] taught a robot to act according to short speech commands in different languages. Increasing the number of commands and chaining simple behaviors to complex ones are of great challenge, especially for real robots.

In this paper, we report our recent work on action chaining, a function based on second order conditioning, which enables the SAIL robot to learn a chain of basic actions through the interactions with a human teacher. Section 2 introduces the method. Section 3 discusses how to autonomously move up to higher abstraction, although not necessarily symbolic. The experiments are discussed in Section 4.

2 A Double-tree System

Action chaining is a challenging problem that machine learning researchers face in studying the conditioning protocols [8]. Suppose there are two basic actions A_{s1} and A_{s2}, and one composite action A_c which consists of action A_{s1} followed by action A_{s2} ($A_{s1} \rightarrow A_{s2}$). In the teacher's mind, these three actions correspond to the voice commands, C_{s1}, C_{s2}, and C_c, respectively. Suppose an agent has learned to conduct A_{s1} and A_{s2} upon hearing voice commands C_{s1} and C_{s2}, i.e., $C_{s1} \rightarrow A_{s1}$ and $C_{s2} \rightarrow A_{s2}$. How can the agent learn to conduct A_c upon hearing voice command C_c only? Mathematically, an action chaining problem can be written as the following A.P.,

$$C_c \rightarrow C_{s1} \rightarrow A_{s1} \rightarrow C_{s2} \rightarrow A_{s2} \Rightarrow C_c \rightarrow A_{s1} \rightarrow A_{s2} \quad (3)$$

A.P. (3) shows that the action chaining problem is a chained secondary conditioning problem, with C_{s1} and C_{s2} as the triggering CS and $C_c \rightarrow A_{s1} \rightarrow A_{s2}$ as the newly established chained stimulus-response association. As an example, shown in Fig. 1 are the gripper tip trajectories we have trained our SAIL robot to make. The problem of action chaining here is to enable

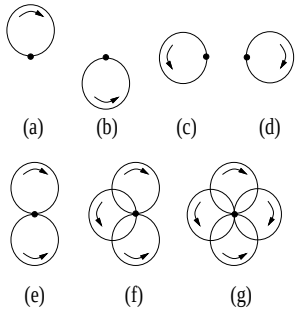

Figure 1: Gripper tip trajectories of the SAIL robot. (a)-(d) are basic actions, upper circle, lower circle, left circle, and right circle, each of which starts from the black dot. (e)-(g) are composite actions chaining some or all of the basic ones.

the robot make complicated curves upon acquisition of basic ones, through more interactions only.

Although we present the chaining problem in a symbolic way above, it is important to note that each symbol here (Cs and As) represents a stochastic process. To be correct, each symbol corresponds to many possible sequences of random vectors representing sensory inputs,

$$C = \{C(1), C(2), \ldots, C(t)\}$$

or effector signals,

$$A = \{A(1), A(2), \ldots, A(t)\}.$$

At each time instance, sensory inputs are collected by real sensors (vision, audition, and taction) and represented in its original numerical format, which makes the dimension of the vectors as high as a few hundreds (for audition) to a few thousands and above (for vision). With the same sampling rate, time varying vectors of control signals are sent to the motor controllers. Depending on the number of motors, the vector of control signals may vary from 1 to 13 in the case of our SAIL robot. Dealing with numerical representations enables the same program to learn new stimuli and behaviors without reprogramming intervention. However, it also poses greater challenges than dealing with symbolic representations. To keep the discussion clear, we will still use a symbolic notation to denote a random process: sequences of random vectors.

A challenge of using numerical representation is to efficiently manipulate high dimensional signals. We used a technique called Incremental Hierarchical Discriminant Regression (IHDR) [10]. The tree structure of IHDR incrementally establishes a mapping,

$$p(t+1) = g(s(t))$$

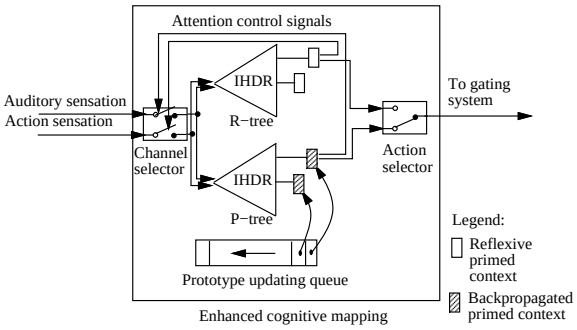

Figure 2: A double-tree system.

from the input (the last context $s(t)$) to the output (the primed context $p(t)$), and continuously improves the mapping in real-time. The last context includes the perceptual information such as audition and motor position. The primed context p, consists of three parts, a primed sensation vector p_s, a primed action vector p_a, and a value Q associated with the primed action p_a.

2.1 R-tree and P-tree

As shown in A.P. (3), the challenge of action chaining comes from missing the contexts, the commands C_{s1} and C_{s2}. To handle the missing context, we designed a double-tree system (Fig. 2). The major difference of this system from the one we used in [11] is that we added a tree with a *prototype updating queue* (PUQ) into the cognitive mapping module. We call the original tree the reality tree, or *R-tree*, and the added one the priming tree, or *P-tree*. P-tree looks ahead (priming) but R-tree does not. The goal of PUQ for P-tree is to enable looking ahead (priming) in the future by doing sensation and value backpropogation while experience occurs now. The PUQ maintains a list of pointers to the primed contexts retrieved by the P-tree. At every time instance, a pointer to a newly retrieved primed context enters the PUQ while the oldest one moves out. When the pointers are kept in PUQ, the primed contexts they point to are updated with a recursive model adapted from Q-learning [9],

$$p^{(n)}(t) = p^{(n-1)}(t) + \frac{1+l}{n}(\gamma p^{(n-1)}(t+1) - p^{(n-1)}(t)) \quad (4)$$

where, $p^{(n)}(t)$ is the primed context at the time instance t, n represents the number of times $p^{(n)}(t)$ has been updated, and γ is a discount rate. l is an amnesic parameter introduced by us, which is typically positive and is used to give more weight on the newer data points. Reorganizing Eq. (4), we have,

$$p^{(n)}(t) = \frac{n-1-l}{n}p^{(n-1)}(t) + \frac{1+l}{n}\gamma p^{(n-1)}(t+1) \quad (5)$$

which shows more clearly that a primed context $p^{(n)}(t)$ is updated by averaging its last version $p^{(n-1)}(t)$ and the time-discounted version of the current primed context $p^{(n-1)}(t+1)$. In this way, the information embedded in the future context, $p^{(n-1)}(t+1)$ in model (4), is recursively back-propagated into earlier primed contexts. Therefore, Eq. (4) is effectively a prediction model. When an earlier context is recalled, it contains expected future information. Here, the power of vector representation is clear.

2.2 Algorithm

At each time instance, the developmental program executes the following learning procedure[1]:

1. Collect the sensations from both the auditory sensor, x_s, and the action sensor, x_a.
2. The channel selector forms a single sensation vector, $s(t) = (x_s, x_a)$, by replacing x_s or x_a with a zero vector, depending on the attention control signal.
3. Both the R-tree and the P-tree learn $s(t)$.
4. Two primitive prototypes, high dimensional vectors $P_R(t)$ and $P_P(t)$, are retrieved from the trees. They are best matched prototypes.
5. If there is an imposed action $a_i(t)$, find the primed contexts, $p_R(t)$ and $p_P(t)$ associated with $P_R(t)$ and $P_P(t)$, respectively, which contain the similar primed action part at $a_i(t)$, and set their values to be 1.
6. Compute the confidence index of the primed context for both trees and select the one with higher confidence, whose primed action would be the external action, $a_e(t)$. Its associated value is termed as $v_e(t)$.
7. The pointer to the primed context $p_P(t)$ enters the PUQ and each entry in the PUQ is updated according to model (4). Specifically, the sensation part is updated using $P_P(t)$, the action part using $a_e(t)$, and the value part using $v_e(t)$.
8. Send $a_e(t)$ to the corresponding effectors.

[1] More details are available in [12].

2.3 Training procedure

Before learning the composite action, the agent should learn to perform basic actions. So, the whole training procedure is divided into two steps: (1) The trainer teaches the agent, through physical interactions, to conduct basic actions following voice commands; (2) The trainer instructs the agent through voice commands to conduct the composite action. Our real-time system trained by 63 users learned to establish the composite action with a correct rate of about 93%. For details about the experiments and discussion, the reader is referred to [12].

3 A Multi-level System

3.1 Learn to adjust behaviors

The above described system learns the basic actions through supervised learning. While supervised learning has the advantage of efficiency, the system is not very flexible in adjusting behaviors. For example, suppose the robot has been taught to lift its arm when hearing the command "one". Later on, the teacher does not want the robot to lift its arm any more, which is known in psychology as "extinction". There is no way to impose a "do-nothing" action. To increase the system's behavior adjustment capability, we need to add a reinforcement learning mechanism into the system.

The first attempt of adding a reinforcement learning mechanism was straightforward. We simply fed reinforcement signals to the system and modified the prediction model for updating Q into the following one,

$$\begin{aligned} Q^{(n)}(t) &= Q^{(n-1)}(t) \\ &+ \frac{1+l}{n}[r(t) + \gamma Q^{(n-1)}(t+1) - Q^{(n-1)}(t)] \end{aligned} \quad (6)$$

In this way, we can tune the Q value in the primed context through the reinforcement signals and, consequently, we may tune the behavior of the system. The good news is that we do not need to change other steps in the algorithm of Section 2.2. As can be seen, P-tree is able to prime actions that fit the current context but not exactly (for dealing with the missing context issue discussed before). Which alternative actions from the two trees are adopted is determined by their primed values.

However, this straightforward strategy did not work out well. Because of the prediction model (4) and the PUQ, the primed context of a certain primitive prototype would be propagated to others so that multiple

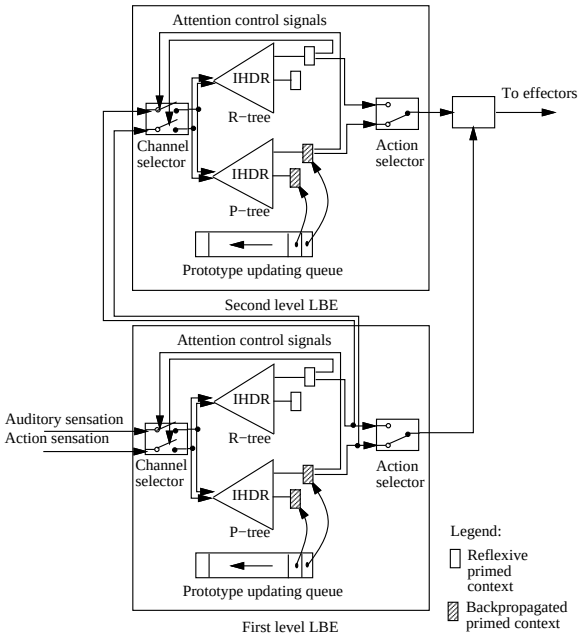

Figure 3: A two-level architecture of SAIL.

primitive prototype may have similar primed context, especially its primed action part. In the real world training situation, the exact context is not guaranteed to be repeated, which means not all the propagated primitive prototypes may show up in the following training session. As a result, while we may adjust the primed context of certain primitive prototypes, we may not be able to adjust all the propagated ones. This is the well-known "abstraction" issue.

To solve this problem, we designed the architecture shown in Fig. 3. We call the whole module of Fig. 2 as the level building element, LBE. The new architecture has two levels of LBEs. All the LBEs run the algorithm of Section 2.2 except that the second-level one takes the primed sensations of the lower-level one as the input. The two-level system works like this. Because of the prediction model (4), the primed sensation vectors are the averaged version of its future context vectors. This means that, the vectors which used to fall into different primitive prototypes in the first level LBE L_1 may be grouped into one primitive prototype of the second level LBE L_2, as shown in Fig. 4. As a result, even though it may take long before a particular primitive prototype is revisited in L_1, L_2 would cover the context it represents.

For making decisions on choosing actions, L_2 checks the Q value of the action that is closest to the one selected by L_1. If it is larger than zero, the corresponding control signals would be sent to the effectors. Oth-

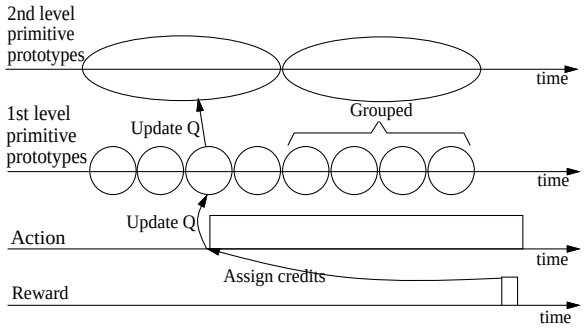

Figure 4: Internal mechanism of the two-level architecture.

erwise, the control signals would be blocked. In other words, no action would be taken. The Q value given by L_2 will enter L_1 as an internal reward so the behavior of L_1 will be tuned even if there is no constant environment feedback.

In the current implementation, the R-tree of L_2 was not used. To quickly assign the reward to the right context, a credit-assignment criterion is preprogrammed by targeting the reward to the starting point of a recently conducted action, as shown in Fig. 4. Issues on automatic credit-assignment need further investigation.

3.2 Algorithm

As a summary of the last section, the following is the learning algorithm of the two-level system at each time instance.

1. Collect the sensation from both the audio sensor, x_s, and the action sensor, x_a.

2. For L_1, do step 2 through 5 of the algorithm of Section 2.2. Denote the primitive prototypes retrieved by R-tree and P-tree as $R_{P1}(t)$ and $P_{P1}(t)$, respectively. Find the primed context with highest confident index among the primed contexts associated with $R_{P1}(t)$ and $P_{P1}(t)$ and denote it as $p_1(t)$.

3. Take the primed sensation part of $p_1(t)$ as the input to L_2 and do step 2 through 5 of the algorithm of Section 2.2 for L_2. Denote the primitive prototype retrieved by P-tree as $P_{P2}(t)$. Find the primed context of $P_{P2}(t)$ with the primed action part most similar to that of $p_1(t)$ and denote it as $p_2(t)$. Let $p_2(t)$ enter the PUQ of L_2 and update according to model (4). For updating Q, use model (6).

Figure 5: SAIL robot house-made at Michigan State University.

4. If the Q value of $p_2(t)$ is larger than zero, send the primed action part of $p_1(t)$ to the corresponding effectors. Otherwise, send the zero vector to the corresponding effectors.

5. Let $p_1(t)$ enter the PUQ of L_1 and update according to model (4). For updating Q, use

$$\begin{aligned} Q^{(n)}(t) &= Q^{(n-1)}(t) + \frac{1+l}{n}[r(t) + Q_2^{(n-1)}(t) \\ &+ \gamma Q^{(n-1)}(t+1) - Q^{(n-1)}(t)] \end{aligned}$$

where $Q_2^{(n)}(t)$ is the Q value of $p_2(t)$.

From the above discussions, we can see that the role of L_2 is effectively a prediction and evaluation system which evaluates the behaviors of L_1. This gives our system have some characteristics of the well-known actor-critic methods [7, page 151]. The difference here are (1) our method starts from raw sensors instead of symbolic input, (2) both the "critic" L_2 and the "actor" L_1 learn from the environment, (3) the critic is not task-specific. It gives the system a quicker response to the change of the environment by not letting the actor wait until the critic realizes the change. Another interesting thing about our system is that the "critic" and the "actor" have similar architecture and learn in the same manner, which saves a lot of design effort. L_2 may be related to higher-order cortex in biology but this paper does not address this aspect.

4 Experimental Results on SAIL

In this section, we report some experiments conducted on our SAIL robot. We interacted with the

SAIL robot through its audition sensor, a microphone, and the micro-switch sensors on its arm[2]. One of the switch sensors were defined as a biased sensor to accept reinforcement signals. Specifically, if the reading of that biased sensor was 1, the reward was 1, and if the reading was -1, the reward was -1. Other two switch sensors were used to impose the four basic arm actions, which let the gripper tip make the four circles shown in Fig. 1.

4.1 Behavior establishment

In this experiment, we taught the robot basic actions. It is the first step of action chaining. The training procedure goes as follows, (1) The trainer speaks one of the voice commands ("one", "two", "three", and "four" in our experiment); (2) At the end of the utterance, the trainer presses the switch sensor to impose one the four actions; (3) Wait for a couple of seconds and go back to (1).

The trainer might need to repeat some of the commands. But according to our observation, within three repeats, a single behavior can be established with correct rate of 95-100%.

4.2 Behavior extinction

In this experiment, we want to adjust the established behaviors of the robot. Suppose the robot has learned to do the upper-circle movement after hearing "one" and, later, we do not want it to move the arm after command "one". The training procedure goes as follows, (1) The trainer speaks "one"; (2) If the robot moves its arm, the trainer presses the biased switch sensor to give punishment; (3) Wait for a couple of seconds and go back to (1).

To trace the internal changes of the robot, we saved the information related to decision making at every time step which was an interval of about 20 millisecond. The upper panel of Fig. 6 shows the sound volume along the time axis. The vertical dash lines show the time instances when the robot started the upper-circle movement. Among them, the first one is when the trainer imposed the action. The lower panel of Fig. 6 shows the changes of Q values of two actions associated with a particular primitive prototype. A non-action means the robot stays still and action 1 means upper-circle movement. In the real-time experiment, there were typically more than one thousand primitive prototypes saved in each IHDR tree. We

[2]For hardware specifications of SAIL, the reader is referred to [11].

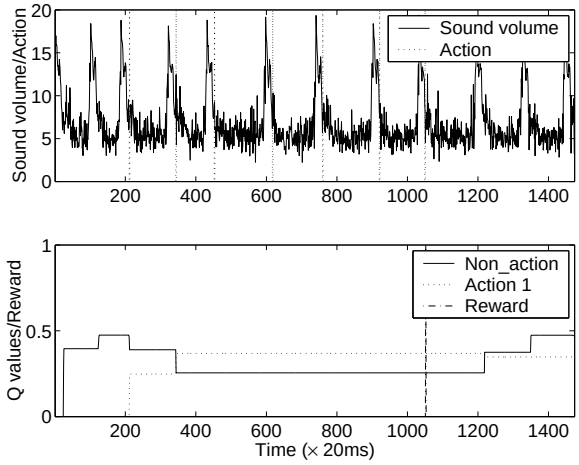

Figure 6: The behavior changes of SAIL with the two-level architecture.

were only able to get a hold of a particular primitive prototype when it was retrieved. For the time instances when the particular primitive prototype was not retrieved, we simply showed its Q value when it was retrieved previously. That is why the reader can not see frequent changes of Q in the figure. This is also the reason that the changes of the Q values show up much later than the reinforcement signals. However, we do see the Q value of action 1 starting to grow after the action was imposed. At the same time, that of the non-action dropped. This means the robot gradually preferred action 1 to the non-action. After receiving a punishment reinforcement signal at the time instance shown by the vertical dash line in the lower panel of Fig. 6, the robot's preference reversed by the changing Q values. From the upper panel, we can see that the previous established behavior was extinct thereafter.

For comparison, we did the same experiment on a single level system. As shown in Fig. 7, although the system can establish the actions, it took a much longer time to adjust its behavior to the desired performance.

4.3 Action chaining

In this experiment, we teach the robot to do the composite action. The training procedure goes as follows, (1) A trainer speaks "start", followed by "one", "two", "three", "four"; (2) Wait for a couple of seconds and go back to (1). This is essentially a process that the trainer teaches the system to do the composite action through verbal instructions. After repeating the procedure several times, the trainer speaks only "start". If the robot continuously do the four actions, we count it as a success.

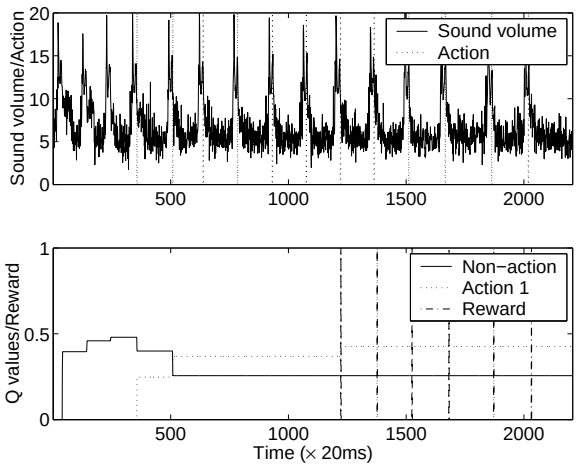

Figure 7: The behavior changes of SAIL with the one-level architecture.

We repeated the experiment of Section 4.1 followed by this experiment for 20 times. Table 1 shows that there was only one case when the chaining of four actions failed. In that particular case, the robot confused the command "start" with the command "three" because their beginning portion sounded similar. Even in this case, the chaining of the first two actions was still successful.

Table 1: Performance of SAIL doing action chaining

No. of Actions	1	2	3	4
C.R (%)	100	100	95	95

4.4 Execution time and memory size

The speed is a very important issue for a real-time system. In each time step, the SAIL robot needs to collect the sensory data from a sound card as well as the touch and pressure sensors, compute the mel-frequency capstrum coefficients (MFCCs) for each speech frame, update and retrieve the IHDR trees, and submit the control signals to the controllers of the effectors. As each speech frame covers 200 points and the auditory digitization frequence is 11.025 kHz, the above computations must be completed in some 18.1ms, which is very tight. Typically, the more complex the sound distribution is (with fewer repeated patterns), the larger the trees are. And larger trees mean longer execution time although the IHDR has a logarithmic time complexity. To push the system to its extreme, we played loud music to it and recorded the execution time of above computations within each time step.

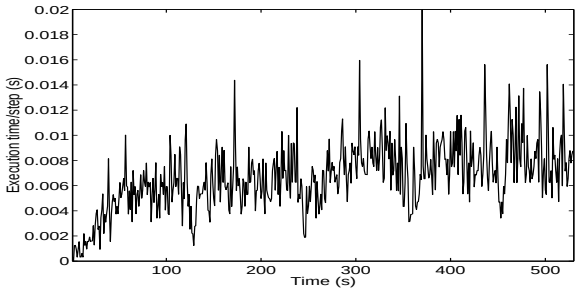

Figure 8: The average execution time for each time step.

Fig. 8 shows that the execution time tended to grow at the beginning and it flattened out after about 100 second. Overall, the execution time is well under the interval of 18.1ms, though it varies all the time because a primitive prototype may be stored at and retrieved from different depths of the trees. The total size of the three trees reached about 300MB after 550 seconds of "music listening". While the growing speed of the trees is still an issue, the good news is that the system was still running smoothly. To provide an idea about the structure of the trees, some data about the resulted IHDR trees is shown in Fig. 9, where the horizontal axes are the depth of the tree and the vertical axes are the number of nodes or the number of primitive prototypes saved. The P-tree of L_2 is significantly smaller than the R-tree and the P-tree of L_1 because of the "grouping" we mentioned in Section 3.1.

5 Conclusions

We proposed a developmental architecture for an agent which is able to learn chained actions through chained secondary conditioning, as well as its algorithm. Our experiments showed that, with this architecture, a robot could successfully develop more complex behaviors (chained actions) upon the acquisition of basic ones. The scheme allows the teacher to change reward criteria, useful for behavior shaping when the agent moves to more experienced age groups.

As far as we know, the work reported here is the first in achieving action chaining in uncontrolled environment from raw sensory signals without knowing perception and action classes, not even the number of classes. Architecture design and the power of IHDR played an important role in the success. Our next step is to test this architecture with more sensing modalities and effectors.

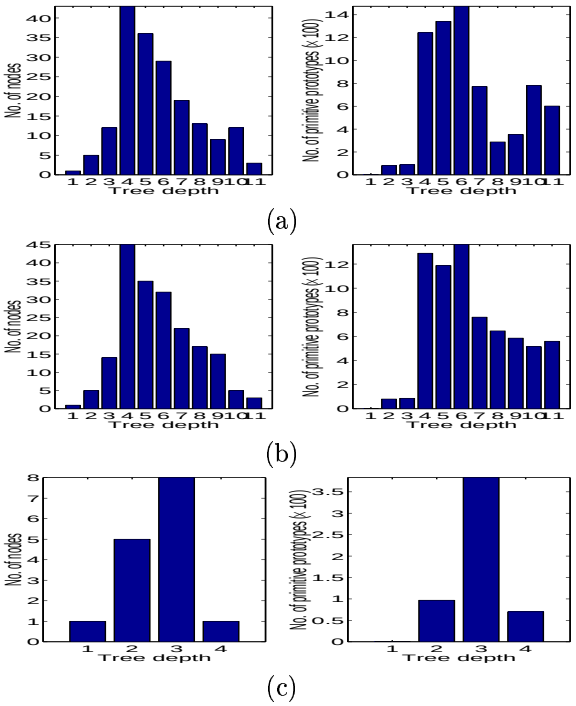

Figure 9: The node distribution and primitive prototype distribution in the trees: (a) R-tree of the first level LBE; (b) P-tree of the first level LBE; (c) P-tree of the second level LBE.

References

[1] C. Balkenius and J Moren. Computational models of classical conditioning: a comparative study. *Lund University Cognitive Studies*, 62, 1998.

[2] M. Domjan. *The Principles of Learning and Behavior*. Brooks/Cole, Belmont, CA, fourth edition, 1998.

[3] Q. Liu, S. Levinson, Y. Wu, and T. Huang. Robot speech learning via entropy guided LVQ and memory association. In *Proc. INNS-IEEE International Joint Conference on Neural Networks*, pages 2176–2181, Washington, DC, July 14-19, 2001.

[4] D. Roy, B. Schiele, and A. Pentland. Learning audio-visual associations using mutual information. In *Workshop on Integrating Speech and Image Understanding, Proc. Int'l Conf. Comp. Vision*, Corfu, Greece, September, 1999.

[5] L.M. Saksida, S.M. Raymond, and D.S. Touretzky. Shaping robot behavior using principles from instrumental conditioning. *Adaptive Behavior*, 22(3/4):231–249, 1998.

[6] O. Sporns, N. Almassy, and G.M. Edelman. Plasticity in value systems and its role in adaptive behavior. *Adaptive Behavior*, 7(3), 1999.

[7] R.S. Sutton and A.G. Barto. *Reinforcement Learning – An Introduction*. The MIT Press, Chambridge, MA, 1998.

[8] D.S. Touretzky and L.M. Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5(3/4):219–247, 1999.

[9] C. J. Watkins. Q-learning. *Machine Learning*, 8:279–292, 1992.

[10] J. Weng, W.S. Hwang, Y. Zhang, et al. Developmental humanoids: humanoids that develop skills automatically. In *Proc. The First IEEE-RAS International Conference on Humanoid Robots*, Boston, MA, September 7-8, 2000.

[11] Y. Zhang and J. Weng. Grounded auditory development of a developmental robot. In *Proc. INNS-IEEE International Joint Conference on Neural Networks*, pages 1059–1064, Washington, DC, July 14-19, 2001.

[12] Y. Zhang and J. Weng. Chained action learning through real-time interactions. In *Proc. INNS-IEEE International Joint Conference on Neural Networks*, Honolulu, HI, May 12-17, 2002.