LEARNING BEHAVIOR SELECTION THROUGH INTERACTION BASED ON EMOTIONALLY GROUNDED SYMBOL CONCEPT

TSUTOMU SAWADA
Information Technologies Laboratories, Sony Corporation,
6-7-35 Kitashinagawa Shinagawa-ku, Tokyo, 141-0001, Japan
tsawada@pdp.crl.sony.co.jp

TSUYOSHI TAKAGI
Entertainment Robot Company, Sony Corporation,
5-11-3 Shinbashi Minato-ku, Tokyo, 105-0004, Japan
takagi@erc.sony.co.jp

YUKIKO HOSHINO
Life Dynamics Laboratory Preparatory Office, Sony Corporation,
6-7-35 Kitashinagawa Shinagawa-ku, Tokyo, 141-0001, Japan
yukiko@pdp.crl.sony.co.jp

MASAHIRO FUJITA
Information Technologies Laboratories, Sony Corporation,
6-7-35 Kitashinagawa Shinagawa-ku, Tokyo, 141-0001, Japan
mfujita@pdp.crl.sony.co.jp

In this paper, we propose a learning algorithm for the action selection mechanism in the EGO architecture, which is designed for autonomous behavior control of a humanoid robot. The concept of behavior value is introduced for action selection. The behavior value of each behavior module depends on external stimuli and internal states, and the behavior module with the highest behavior value is selected according to the situation. We address the importance of learning the behavior value of each behavior. We describe how to compute behavior values for behavior modules through interaction with humans and environment. We implemented the learning algorithm on QRIO SDR-4X II, a small humanoid robot, and confirmed that for a given interaction-driven behavior module, a high behavior value is obtained when interacting with a friendly user. A similar result is obtained for a proper color painted ball for the soccer play behavior module.

Keywords: EGO architecture, behavior value, learning, QRIO SDR-4X II.

1. Introduction

We have previously described the autonomous behavior control architecture, the EGO Architecture, for consumer entertainment applications[1]. For the purpose, we developed a small humanoid robot QRIO SDR-4X (hereinafter QRIO). It is required for such a robot to walk around in a home environment, to respond to social cues and other stimuli, to find and identify users, and to communicate with users naturally. There are many embedded technologies in the robot, such as real-time dynamic walking control, map-building of the environment, human detection and identification, speech recognition and synthesis, and natural language processing for verbal communication. Most of all, it is important for the robot to

behave spontaneously and naturally and the EGO architecture is developed for such purposes.

From a Behavior Control Architecture point of view, proper behavior coordination is one of the most important issues. In many Behavior Based architectures[1], behaviors are controlled by so called “releasers”, which are carefully designed and debugged by a human. Usually, the releasers are described by a TRUE-FALSE logic table, and one releaser that evaluates to TRUE activates the corresponding behavior in the situation[2].

In EGO architecture, we assign a “behavior value” to each behavior module, and behaviors are coordinated based on that value. The behavior value could be considered similar to a Q-value in reinforcement learning, where the action with the highest Q-value is selected to get the higher reward. In a similar way, the behavior with the highest behavior value is selected to regulate the internal variables. The details of this process are described later in this paper, but in short the internal variables must be regulated to remain within certain ranges. This is a key factor for autonomous or spontaneous behavior in the EGO architecture.

Returning to the action selection issue, since a releaser is programmed manually, the behavior value is also usually programmed or assigned manually. However, if many behaviors are added and the robot acts in a real world environment, it is difficult to determine these behavior values manually. Moreover, in some cases it is impossible to determine the behavior values accurately before the robot actually interacts with the environment. For example, if there are a user (USER-A) who likes to interact with the robot and another user (USER-B) who doesn’t, the robot should determine the behavior value of the interaction behavior module in such a way that a higher behavior value for USER-A and a lower behavior value for USER-B are set. These values cannot be assigned before the robot actually interacts with users.

We already presented the Emotionally Grounded Symbol concept[3, 4] in previous works, where symbols are grounded to an emotional system.

In the remainder of this paper, we first describe an overview of the EGO architecture, followed by the method used to compute the behavior values and how to properly coordinate behaviors based on those values. Then we describe how to learn the behavior values through interactions with the environment. We also describe several implementations and present the results of experiments using QRIO. Then, we review related works and discuss some features of the EGO architecture with respect to the learning mechanism.

Since the EGO architecture is inspired by ethological studies[5], we often use terminology from animal ethology to describe robot`s behaviors; for example, we call “EAT” a battery charge behavior and we call “NOURISHMENT” an internal variable that corresponds to battery charge level.

We should also note that in literature “action selection” and “behavior selection” are often used interchangeably. In the following, we choose the term “behavior selection” because “action” has a more primitive meaning than “behavior”. However, when we refer to other articles, we generally try to use the original

terminology.

2. EGO ARCHITECTURE OVERVIEW

In this section, the individual software components of the EGO Architecture are briefly explained. Fig. 1 provides an overview. Please refer to the paper for more details on the EGO Architecture[6].

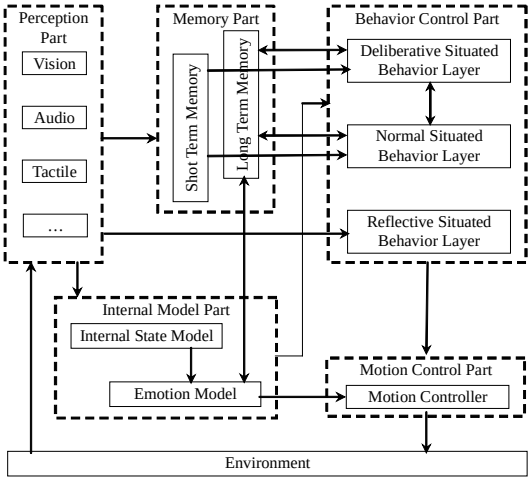


Fig. 1 Overview of the EGO Architecture

2.1. *Short Term Memory (STM)*

STM integrates the results of perception. From audio perception, STM receives the result of not only speech recognition but also sound source direction by multi-microphone localization. Regarding vision perception, STM can store and provide the result of face recognition with its associated direction and distance computed from stereovision. In the case that both, audio and visual directions, are the same, STM merges the results to indicate that they are from the same user.
STM can also compute the relative positions of detected objects (face, ball, etc.) through kinematics. Thus STM can store and recall results located outside of the limited view range.

2.2. *Long term memory (LTM)*

LTM associates the recognition results with an internal state. For example, LTM can associate an acquired name with an identified object or an identified voice, and

change the internal state associated with the particular target object[8].

2.3. *Internal state model (ISM)*

ISM maintains various internal state variables. It alters their values depending on the passage of time and incoming external stimuli. Basically, a behavior module is selected in order to keep these internal state variables within proper ranges. ISM is the core for spontaneous behavior and response generation to external stimuli.

2.4. *Emotion model (EM)*

EM has 6+1 emotions: ANGER, DISGUST, FEAR, JOY, SADNESS, SURPRISE, and NEUTRAL. They are based on Ekmann's proposal[7]. Each emotion has an associated value[8].

2.5. *Situated behavior layer (SBL)*

The Behavior control part is organized into three SBL modules, D-SBL (Deliberative SBL), N-SBL (Normal SBL) and R-SBL (Reflexive SBL). D-SBL realizes the behavior control for deliberative behavior, N-SBL realizes the behavior control for homeostatic behavior, and R-SBL realizes the behavior control for quick responses.

Each SBL controls selection and activation of behavior modules. Each behavior module has two basic functions: Monitor and Action. Monitor function periodically and concurrently creates a value, which is called the behavior value (*BV*), using internal state variables and external stimuli. It indicates how relevant the behavior is for the situation (e.g., observing an object, a sound event, etc.). The details of this computation are described below.

Behavior selection is based on the *BV*s either by a Greedy method, where a maximum *BV* is selected, or by a soft-max policy, where a larger *BV* is selected with larger probability. Selected behavior modules are given execution permission.

Availability of necessary resources for execution, e.g., head, arm, speaker, etc., are also considered in the competition. In the case where there is no resource conflict among behavior modules, all of them are given execution permission and they execute concurrently.

After a behavior module is granted execution permission, the Action function actually performs the behavior; it is implemented as a state machine. Each node can output, for example, a motion command (designed motion command, walk command, tracking command, etc.) and can decide state transition depending.

Figure 2 shows a behavior module and the associated process.

A tree structure is used to organize the behavior modules. An abstract behavior can

be divided into concrete sub-behaviors. For the example, as shown in Fig. 3, “Soccer” can be decomposed into “Search ball”, “Approach ball” and “Kick ball”. Also “Approach ball” can be decomposed into “Go to ball by walk”, “Track ball by head”, “Speak for approach”, etc.

In the parent behavior module in the tree structure, a monitor function can also determine the *BV* considering the child *BV*s instead of relying only on the internal state variables and external stimuli. The Action function of a parent module can also use a child behavior module instead of a motion command[6, 9].

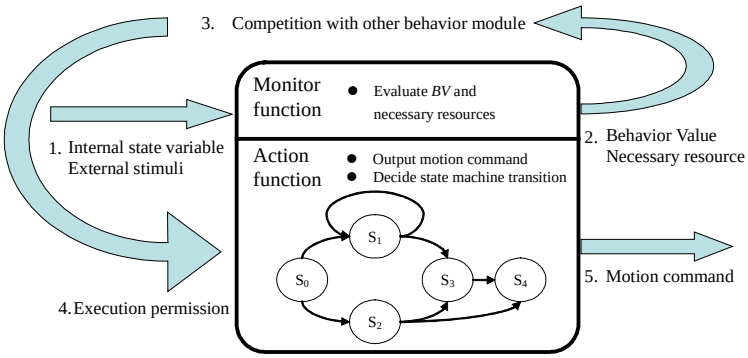


Fig. 2. Behavior module and process

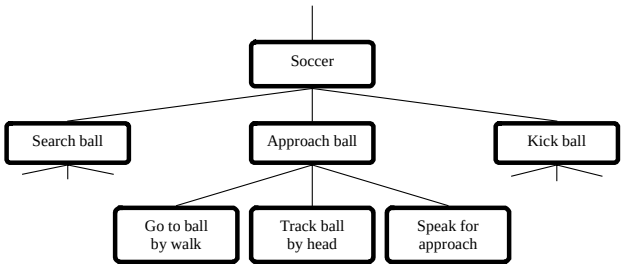
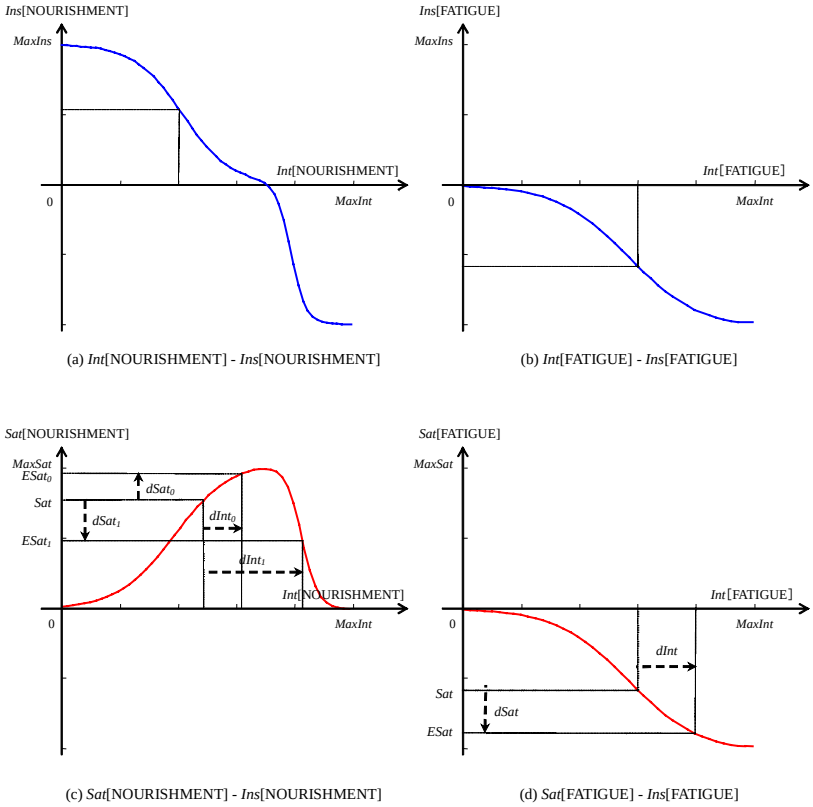


Fig. 3. Tree structure of behavior modules

3. LEARNING BEHAVIOR VALUE

In this paper, we focus on the learning of *BV*’s to realize homeostatic behavior in the Normal Situated Behavior Layer (N-SBL). Performing a behavior causes changes in the internal state variables. Each behavior module evaluates how much the internal state changes as a result of performing the activity. This association is learned in each behavior module. The evaluation and learning of *BV* are described in detail in the following subsection.

3.1. *Evaluation of behavior value*



(a) *Int*[NOURISHMENT] - *Ins*[NOURISHMENT] (b) *Int*[FATIGUE] - *Ins*[FATIGUE]

(c) *Sat*[NOURISHMENT] - *Ins*[NOURISHMENT] (d) *Sat*[FATIGUE] - *Ins*[FATIGUE]

Fig. 4. *Ins*[*i*] - *Int*[*i*] and *Sat*[*i*] - *Int*[*i*] in the behavior module

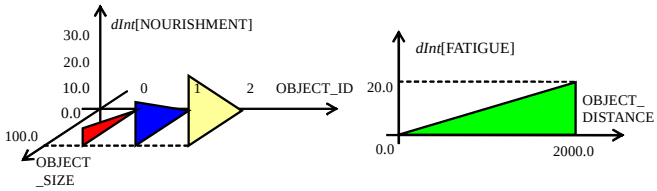


Fig. 5. Database about expected change in the internal state variable

Each *BV* is composed of a motivation value (*Mot*) and a releasing value (*Rel*).
The evaluation of *Mot*, *Rel* and *BV* is described using as an example a behavior to regulate the NOURISHMENT state variable.
From our viewpoint, NOURISHMENT directly depends on the battery charge level and the activity of charging-battery is a pseudo-eating behavior for which the food

source i.e. the object of the activity is the battery station. In the following we will briefly refer to such a behavior using an ethological description e.g. “approach and eat an object”

The motivation value is the degree to which the instinct drives the behavior module. It is derived from internal state variables and is composed of instinct values.

An instinct value (*Ins*[*i*]) is designed for each specific internal state variable (*Int*[*i*]).

Two examples for NOURISHMENT and FATIGUE are shown in Fig. 4 (a) and (b) and can be interpreted as follows. The less nourishment there is, the larger the instinct to eat is. Also, in the case of large amount of nourishment, this instinct turns negative to realize a moderation or reduction in eating behavior (satiety). Fatigue has a negative effect. The greater the fatigue, the lower the value of the instinct associated with it.

Mot is evaluated as shown in Eq. (1).

$$Mot = \sum W_{Mot}[i] \cdot Ins[i] \tag{1}$$

where $W_{Mot}[i]$: Weight of $Ins[i]$

The releasing value is the degree regarding how much an external stimulus would satisfy an internal state as a result of the behavior. It is derived from an internal state variable and the external stimuli and is composed of a satisfaction value and the expected satisfaction value.

A satisfaction value (*Sat*[*i*]) is designed for each specific internal state valuable. Examples for NOURISHMENT and FATIGUE are shown in Fig. 4 (c), (d).

To evaluate the expected satisfaction value (*ESat*[*i*]), the behavior module maintains a database on the expected change in the internal state variable (*dInt*[*i*]) against the result of the behavior for the given external stimuli.

Figure 5 depicts an example where the behavior module expects a change in NOURISHMENT and FATIGUE when an external stimulus (OBJECT_ID, OBJECT_SIZE, and OBJECT_DISTANCE) is obtained. This means that when a target object is found which has OBJECT_ID = 1, OBJECT_SIZE = 100, and OBJECT_DISTANCE = 2000, NOURISHMENT would increase 20 and FATIGUE would increase 20 after approaching and eating the target object.

ESat[*i*] and expected change in satisfaction value (*dSat*[*i*]) are shown in Fig. 4 (c), (d). They are interpreted as follows. When $dInt_0$ is determined by observing an $object_0$, the *dSat*[NOURISHMENT] is positive. On the contrary, when $dInt_1$ is determined when observing another $object_1$, for example whose size is larger than $object_0$, the *dSat*[NOURISHMENT] is negative due to overeating. *dInt* for fatigue is related to the distance of an observed object. The farther the distance is, the more dissatisfaction the agent receives.

Rel is evaluated by Eq. (2).

$$Rel = \sum W_{Rel}[i] \cdot \left(W_{dSat}\, dSat[i] + (1 - W_{dSat})ESat[i]\right) \tag{2}$$

where $W_{Rel}[i]$: Weight of $(W_{dSat}dSat[i]+(1-W_{dSat})ESat[i])$
W_{dSat}: Weight of $dSat[i]$ against $ESat[i]$

The term 'releasing value' is derived from ethological studies. When an animal responds to external stimuli, this is interpreted as the external stimuli releasing a behavior, and is thus called a releasing mechanism. In our approach, a behavior is released by a value derived from external stimuli. This releasing value is considered as an enhanced releasing mechanism.
Finally *BV* is evaluated from *Mot* and *Rel* by Eq. (3).

$$BV = W_{Mot}\, Mot + (1 - W_{Mot})Rel \tag{3}$$

where W_{Mot}: Weight of *Mot* against *Rel*

Note that when there is no external stimulus for the behavior module, *BV* is set to 0, so that the behavior module is never selected.

3.2. *Learning of change in the internal state variable*

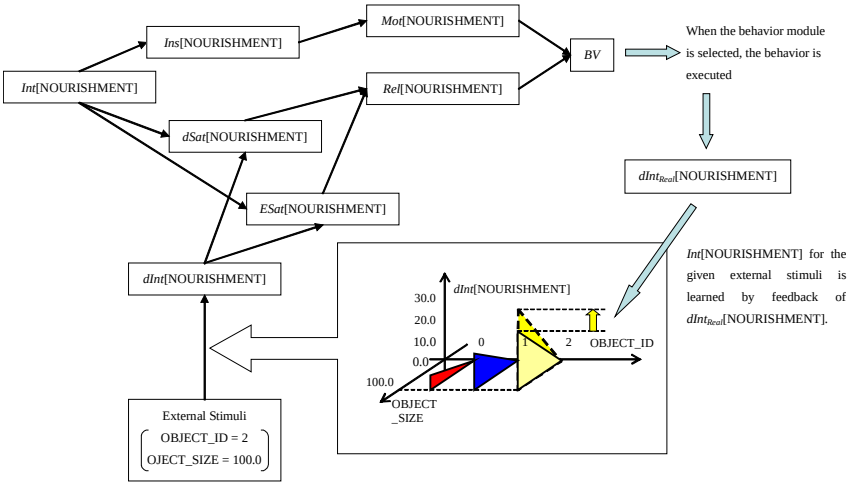


Fig. 6. Process of learning

As mentioned in the introduction, it is difficult to set *BV* properly. It is also important that *BV* changes properly through interactions with the environment.
In the evaluation of *BV*, each behavior module expects $dInt[i]$ based on the database through external stimuli. As a result of the actions of the behavior, the

internal state variable changes. In this paper, *dInt*[*i*] is renewed from feedback of a real change in internal state variables, and the parameters of *BV* are learned.
Figure 6 shows the process of learning using an example of "eat a target object". The behavior module evaluates *BV* from *Int*[NOURSHMENT] and external stimuli OBJECT_ID = 2, OBJECT_SIZE = 100.0 in the database.
Execution of the behavior "eat the target object" results in change in NOURISHMENT ($dInt_{Real}$[NOURISHMENT]). *dInt*[NOURISHMENT] for the given external stimuli is learned by feedback of $dInt_{Real}$[NOURISHMENT] according to the following Eq. (4).

$$dInt[i] \Leftarrow (1-\alpha)dInt[i] + \alpha \cdot dInt_{Real}[i] \tag{4}$$

where α : Learning ratio

For an unknown target object, the default *dInt*[*i*] is set heuristically. Even if the default *dInt*[*i*] is incorrect at first, it will be learned properly because *dInt*[*i*] grounds on real changes in the internal state variables acquired through this process.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

Let us consider two example behaviors for discussion purposes. The first behavior, "Kick a ball" satisfies, for example, VITALITY. A second behavior, "Interact with a user" satisfies e.g. INTERACTION. Two experiments are then conducted. One is to learn the parameters of *BV* through interactions with faces and balls. The other is to have QRIO behave autonomously based on the learned *BV* in the real environment to demonstrate validity from the viewpoint of an entertainment application.

4.1. *Hardware component of QRIO*

Figure 7 shows QRIO's appearance. It is 580 [mm] height, approximately 7 [kg] with battery and possessing 38 DOF. It is a stand-alone robot with three CPUs. The first is for audio recognition and text-to-speech synthesis. The second CPU is used for visual recognition, short- and long-term memory, and the behavior control architecture. The third is dedicated to motion control. Remote processing power and robot control is also available through a wireless LAN.

4.2. *Experimental Implementation*

The tree structure of the behavior modules is shown in Fig. 8. The *Soccer* (*Sc*) sub-tree has three children: *Soccer Search* (*ScSr*), *Soccer Approach* (*ScAp*), and

Soccer Do (*ScDo*).
BV of *Sc* is the maximum *BV* among its children.
ScAp depends on VITALITY and FATIGUE as internal state variables, and BALL_ID and BALL_DISTANCE as external stimuli. Specification of balls is shown in Table 1.
Mot is composed of *Ins*[VITALITY] and *Ins*[FATIGUE], which are shown in Fig. 9 (a) and (b). *Rel* is composed of *dSat*[VITALITY], *dSat*[FATIGUE], *ESat*[VITALITY] and *ESat*[FATIGUE], which are shown in Fig. 9 (d) and (e).
dInt[VITALITY] and *dInt*[FATIGUE] are estimated from BALL_ID and BALL_DISTANCE. Default values for them are shown in Fig. 10 (a) and (b).
The Weight parameters used for evaluation of *BV* are shown in Table 2.
BV is evaluated every hundred milliseconds within each behavior module.

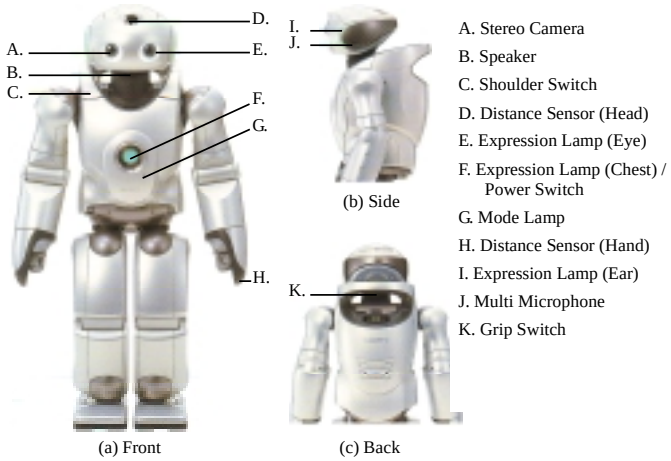


Fig. 7. Appearance of QRIO

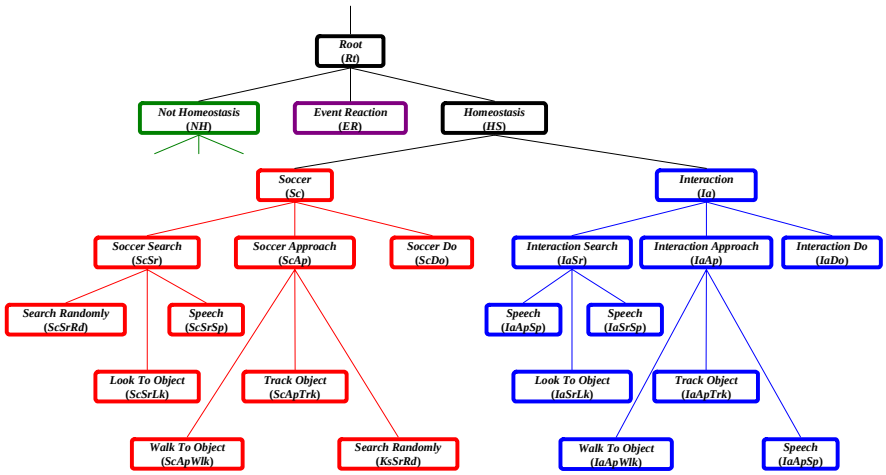
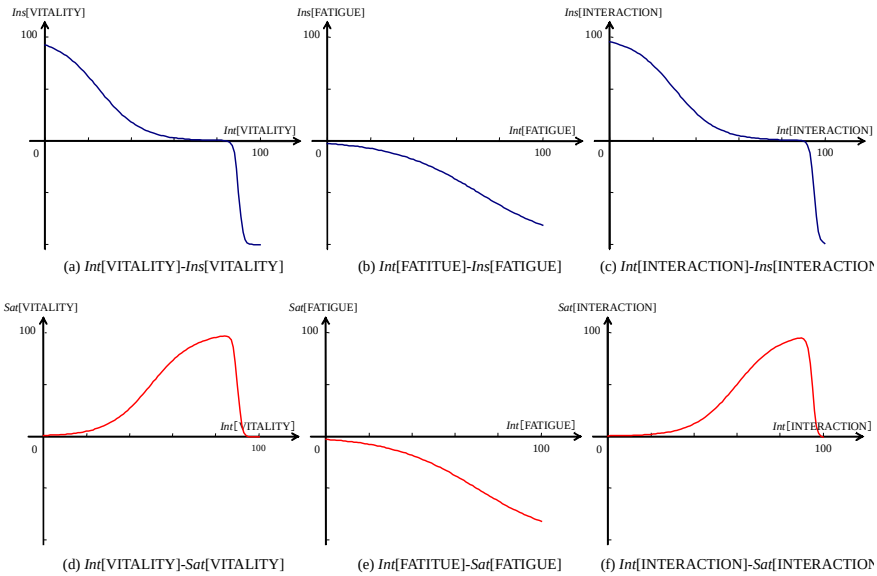


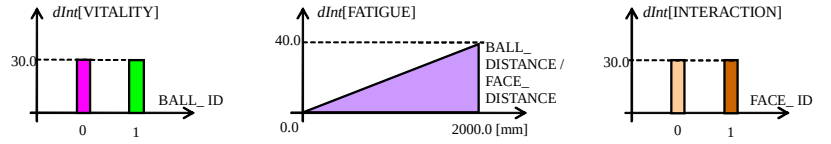
Fig. 8. Tree structure for the application



(a) *Int*[VITALITY]-*Ins*[VITALITY] (b) *Int*[FATITUE]-*Ins*[FATIGUE] (c) *Int*[INTERACTION]-*Ins*[INTERACTION]

(d) *Int*[VITALITY]-*Sat*[VITALITY] (e) *Int*[FATITUE]-*Sat*[FATIGUE] (f) *Int*[INTERACTION]-*Sat*[INTERACTION]

Fig. 9. *Ins*[*i*] against *Int*[*i*] and *Sat*[*i*] against *Int*[*i*]



(a) BALL_ID – *dInt*[VITALITY] (b) OBJECT_DISTANCE – *dInt*[FATIGUE] (c) FACE_ID – *dInt*[INTERACTION]

Fig. 10. Default *dInt*[*i*] against external stimuli

Table 1 Specification of balls

BALL_ID	Color	Radius [mm]	Weight [g]
0	RED	75	330
1	GREEN	75	110

Table 2 Weight parameters for evaluation of *BV*

W_{Mot}	W_{dSat}	W_{Mot}[VITALITY]	W_{Mot}[FATIGUE]	W_{Rel}[VITALITY]	W_{Rel}[FATIGUE]
0.4	1.0	0.8	0.2	0.8	0.2

ScSr depends only on VITALITY and is independent from external stimuli. Evaluation of *BV* is the same as for *ScAp* except for the values of FATIGUE, which

are set to 0.

ScDo depends only on VITALITY as an internal state variable and BALL_ID as external stimuli. Evaluation of *BV* is the same as *ScAp* except for the values of FATIGUE, which are set to 0. If the ball distance is not in the proper range for the kick motion, then *BV* = 0. Note that distance is not used to evaluate *Rel*.

Int[VITALITY] increases proportionally with the distance after kick. It increases by 50 when the distance is 1000 [mm]. $dInt_{Real}$ is evaluated by the difference between the current *Int* and the previous one.

The *Interaction* (*Ia*) sub-tree, composed of *Interaction Search* (*IaSr*), *Interaction Approach* (*IaAp*) and *Interaction Do* (*IaDo*), has the same structure as *Sc* except for internal state variable and external stimuli. INTERACTION and FACE are used instead of VITALITY and BALL respectively.

In the action function of *IaDo*, QRIO requests interaction with the user. When the face comes nearer, an interaction motion command is output and *Int*[VITALITY] increases by 50 (that is $dInt_{Real}$[VITALITY] = 50). On the other hand, if the face does not come nearer for a while, QRIO gives up on interaction. In this case, *Int*[VITALITY] does not increase (that is $dInt_{Real}$[VITALITY] = 0).

Ins[INTERACTION], *Sat*[INTERACTION] and the default value of *dInt*[INTERACTION] are shown in Fig. 9 (c), (f) and Fig. 10 (c) respectively.

In the case that the distance to the detected face is not in the proper range for interaction, *BV* is set to 0.

In our implementation, *dInt*[VITALITY] and *dInt*[INTERACTION] are learned with respect to each target object i.e. BALL_ID and FACE_ID.

Learning ratio α is set to 0.4 in Eq. (4).

The behavior module *Not Homeostasis* (*NH*) does not serve for homeostasis, so its *BV* = 10 constantly. It outputs an idle motion command like leaning the head to one side, tracking a face, etc. When the *BV* of all homeostatic behavior modules are low (all internal states are satisfied), then *NH* is executed.

The behavior module *Event Reaction* (*ER*) does not output any motion command. Instead, when an event triggering a reflexive behavior occurs, then *ER* reserves the same resources required by the reflexive behaviors, sets its BV to a high value i.e. *BV* = 100 and gets activated to prevent a homeostatic behavior module from being selected and executed so not to interfere with the reflexive behaviors.

Table 3 shows the initial conditions of the internal state variables for each experiment. Fig. 11 shows snapshots of the experiment.

Table 3. Initial condition of internal state variables

Experiment No.	*Int*[VITALITY]	*Int*[FATIGUE]	*Int*[INTERACTION]
Experiment 1	80	10	20
Experiment 2	20	10	80
Experiment 3	20	10	80
Experiment 4	20	10	20
Experiment 5	20	10	20

(a) Kick a ball (b) Interact with a user

Fig. 11. Snapshots of the experiment

4.3. *Experiment of learning behavior value*

In Experiment 1, *Ia* sub-tree is active and *Sc* sub-tree is not active because VITALITY is satisfied fully. QRIO tries to search for a user, approach the user, and request interaction with the user. This is executed 10 times for each FACE_ID = 0, 1.

Figure 12 (a), (c) and (e) show the experimental results of learning *dInt*[INTERACTION].

The user with FACE_ID = 0 always accepts QRIO's interaction request while user with FACE_ID = 1 accepts interaction every other time. (Fig. 12 (a)). $dInt_{Real}$[INTERACTION] for each FACE_ID is shown in Fig. 12(c).

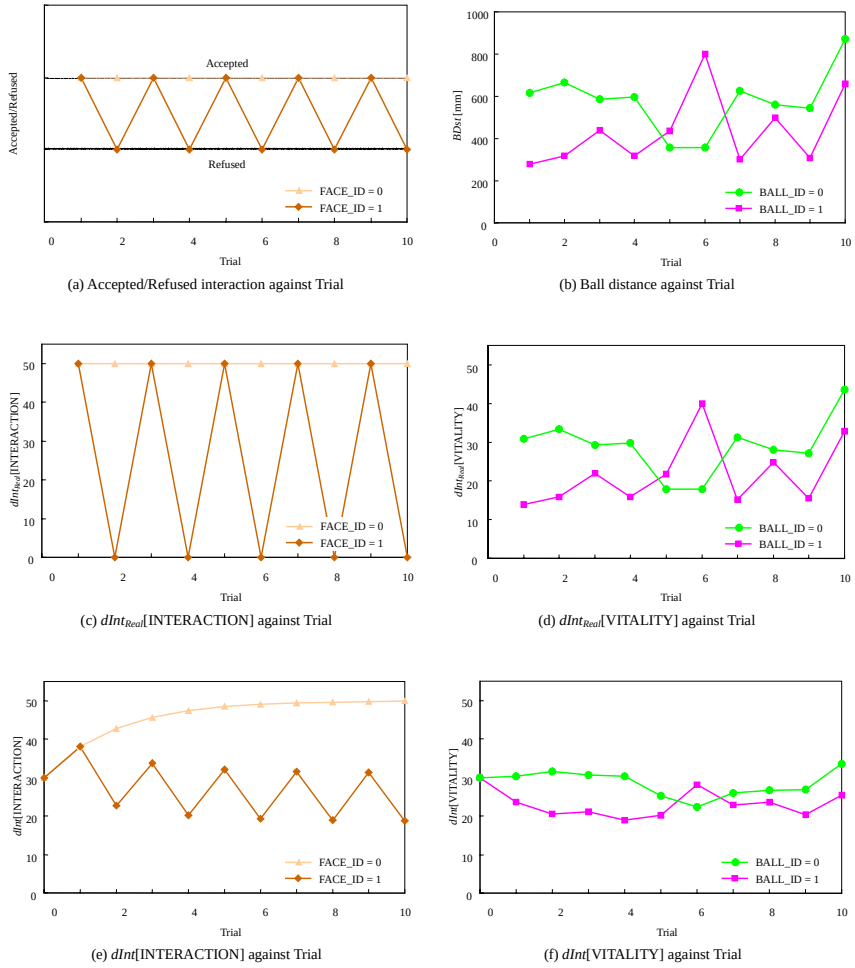
As a result of the learning, *dInt*[INTERACTION] for FACE_ID = 0 gradually converges to $dInt_{Real}$[INTERACTION] = 50.0 and becomes *dInt*[INTERACTION] = 49.9. On the other hand *dInt*[INTERACTION] for FACE_ID = 1 becomes 18.8 with oscillation. (See Fig. 12 (e))

In Experiment 2, the *Sc* sub-tree is active and the *Ia* sub-tree is not active because INTERACTION is satisfied fully. QRIO tries to search for a ball, approach the ball, and then kick the ball. It is also executed 10 times for each BALL_ID = 0, 1.

Figure 12 (b), (d) and (f) show the experimental results of learning *dInt*[VITALITY].

For the results of ball distance, the average is 436.1 [mm] for BALL_ID = 0 and 577.9 [mm] for BALL_ID = 1. This results from the difference in ball weight. Since the ball with BALL_ID = 1 is lighter than the other ball, it travels further when it is kicked. (See Fig. 12 (b)). $dInt_{Real}$[VITALITY] is obtained as shown in Fig. 12 (d) for each BALL_ID.

As a result, the learned values of *dInt*[VITALITY] for BALL_ID = 0 is 25.4 and that of BALL_ID = 1 is 33.6, as illustrated in Fig. 12.



(a) Accepted/Refused interaction against Trial

(b) Ball distance against Trial

(c) $dInt_{Real}$[INTERACTION] against Trial

(d) $dInt_{Real}$[VITALITY] against Trial

(e) *dInt*[INTERACTION] against Trial

(f) *dInt*[VITALITY] against Trial

Fig. 12. Experimental results of learning *dInt*

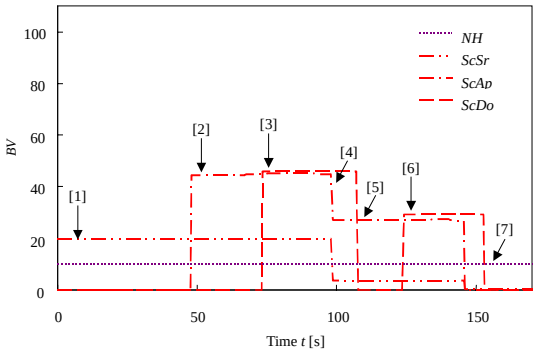
4.4. *Experiment of autonomous behavior based on learned behavior value*

The experimental results of the change in *BV* with learned *dInt* are shown in Fig. 13.

Figure 13 (a) shows the result in Experiment 3, QRIO selects the proper behaviors depending on the detection of a ball and on its relative distance. QRIO is little satisfied with *Int*[VITALITY] in first kick. QRIO kicks the ball again to gain more satisfaction of *Int*[VITALITY]. After the second kick, QRIO is satisfied enough and stops playing soccer: in this situation, QRIO never starts playing soccer again even if QRIO encounters a ball.

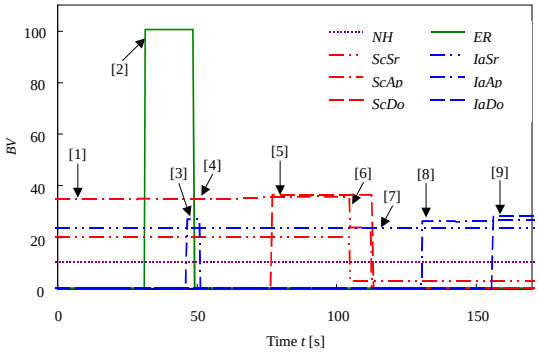
Note that *dInt*[VITALITY] changes from 33.6 to 30.5 after the first kick and from 30.5 to 29.9 after the second kick. Because the proposed learning algorithm is

executed online and in real time, it can then be said that the system has the ability to adapt to the environment.



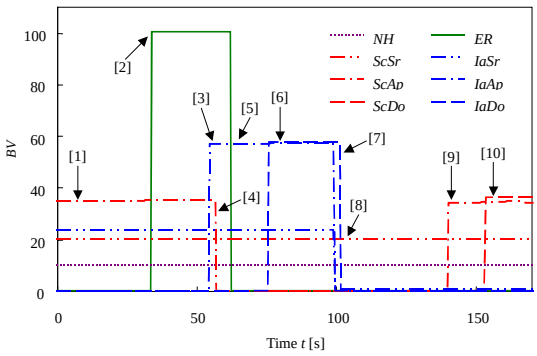
[1] Searches a ball
[2] Finds BALL_ID = 1 and starts to approach it.
[3] Reaches the ball and kicks it.
[4] *Int*[VITALITY] is little satisfied
[5] Approaches the ball again
[6] Kicks the ball again
[7] *Int*[VITALITY] is fully satisfied and *NH* is executed.

(a) Experiment 3



[1] Approaches BALL_ID = 0
[2] Detects clap and looks to the direction
[3] Finds FACE_ID = 1 but ignores the face
[4] Resumes the approaches the ball
[5] Kicks the ball
[6] *Int*[VITALITY] is fully satisfied
[7] Searches a face
[8] Approaches the face
[9] Interacts with the face

(b) Experiment 4



[1] Approaches a BALL_ID = 0
[2] Detects clap and looks to the direction
[3] Finds FACE_ID = 1
[4] Loses the ball
[5] Approaches the face
[6] Interacts with the face
[7] *Int*[VITALITY] is fully satisfied
[8] Searches a ball
[9] Approaches the ball
[10] Kicks the ball

(c) Experiment5

Fig. 13. Experimental results of *BV*

Figure 13 (b) and (c) show results in another experimental condition as described in the following. In Experiment 4, the user with FACE_ID = 1 claps his hands

while QRIO is approaching a ball; QRIO suspends the approach, looks around and finds the user's face. But QRIO ignores the face, resumes the ball approach and kicks it. If we observe the behavior values, we see that *BV*[*ScAp*] is larger than *BV*[*IaAp*].

Figure 13 (c) shows results for a similar scenario but with a different user (FACE_ID = 0). This time (Experiment 5) QRIO suspends its approach to the ball, goes to the user and interacts with him. If we observe the behavior values, we see that *BV*[*IaAp*] is larger than *BV*[*ScAp*].

Our interpretation of Experiments 4 and 5 is as follows

Because of the conditions in which *dInt* parameters are learned (see Experiment 1), expected rewards for selecting a given behavior depend on the environment e.g. in this case on the users as the ball's color doesn't change. Then, In Experiment 5 QRIO prefers interaction with the user to soccer playing as there is a higher expected satisfaction (remember that user with FACE_ID=0 always interact with the robot). On the other hand, in Experiment 4 QRIO does prefer playing soccer to interaction with a user who rarely plays with it (remember that user with FACE_ID=1 does not always interact with the robot). From a different perspective, QRIO develops a bond to sociable users (a user who always interact) and that may in turn attract such users to have further interactions with QRIO.

Finally, it should be noted that the proposed learning algorithm is not for task-oriented domains but rather for entertainment applications. As such, accuracy is not so critical and the observed result can be considered acceptable.

5. RELATED WORK AND DISCUSSION

In this paper, we presented a method for learning parameters used in the calculation of so called behavior values (*BV*). As behavior selection in EGO architecture is based on *BV*, we could argue that this method is an improvement of behavior selection for autonomous robots. Humphrys[11] notes that action-selection algorithms are mainly hand-tuned and little work has been done on learning and action-selection. He proposes reinforcement learning for action-selection and uses a "house robot" for simple tasks e.g. to pick up dirt, return to a base to re-charge and empty its bag, etc. Multiple rewards, each one corresponding to an action, are learned as actions are executed; both predicted and actual rewards are learned thru Reinforcement Learning algorithm. Action-selection mechanism is based on selection of the action with the maximum reward, but there are several alternatives proposed, e.g. selecting the action that maximizes the collection of all rewards. In general Reinforcement Learning research has concentrated on one evaluation function or one goal; however, in real world environments there are many goals that should be considered at the same time. Thus, action selection has to deal with multiple goals in a parallel execution fashion.

The approach described in this paper can also be considered a case of Reinforcement Learning, but we use a regulation mechanism of the internal

variables as the basic mechanism for the rewards: expected changes of the internal variables are learned. The merit of this approach is that the reward values depend on both the system status (internal variables) and the environment (external stimuli). So, even if the environment is suitable for a particular behavior but such behavior is not proper to maintain the internal variables, then the expected reward value is low. On the other hand, even though there may be no suitable external stimuli to trigger a behavior, the motivation value computed from the internal variables may produce an increase of the behavior value and the behavior may be executed. For example, when there is no ball observed, but if the VITALITY is very low, the motivation of the soccer behavior increases so that the corresponding behavior value becomes the maximum one.

Another difference from reinforcement learning lies in the tree structure of behaviors. In our implementation there are many behavior groups, each organized in a sub-tree structure designed manually. While typically in reinforcement learning, action selection is based on time-discounted rewards, since in our system we learn parameters of entire behavior groups, behavior selection evaluates the entire behavior of the behavior group.

In the MOSAIC architecture[12], multiple pairs of predictors and controllers are used. Proper controllers are selected based on the performance of the corresponding predictors. Predictors in MOSAIC can be considered as the Monitor functions in EGO. But in EGO architecture the behavior modules usually perform at a more abstract level than the controller in MOSAIC. As for “motivations” of the behavior, EGO handles multiple motivations based on the regulation rule of the internal variables while in MOSAIC the prediction error can be considered as a general internal variable for the motivation of the behavior.

For example, in our approach, expected changes in internal variable FATIGUE for the approach-a-ball and interact-with-user behaviors might be different because each behavior module has its own database. Also, because of the relationship between satisfaction and internal variable values, different behaviors having the same expected change i.e. *dInt* for a given internal variables may, when executed, cause a different change of the satisfaction value. In other words, the robot may choose the behavior with the highest expected satisfaction even though the expected change of the internal variable is the same.

More generally, similar behavior modules should have similar *dInt* against similar external stimuli. Generalization of the learning result should be considered.

dInt is learned from only a target object as external stimuli. The learning from multi-dimensional external stimuli is one of our future works.

In the current state, learning takes place on a limited part of the system. The remainder is still required to be designed. To clarify the learning-part and designed-part and how to realize other learning-parts are also future work.

6. SUMMARY

In this paper, we describe the learning algorithm of behavior values for the behavior selection problem. The essence of the learning is to make associations of the triples (Behavior, Target, Change of Internal Variables), so that each behavior module can predict the internal variables after the behavior is executed. Then, based on the regulation mechanism of the internal variables each behavior can compute its behavior value in a given situation.
We implemented this algorithm using QRIO, and confirm that the learning results result in different behavior tendencies. For a friendly user, an interaction behavior is often selected, but for an unfriendly user, other behaviors are selected, and so on.

Acknowledgements

We greatly appreciate Dr. Ronald C. Arkin at the Georgia Institute of Technology for his discussion of the architecture, and all researchers and engineers for QRIO in Sony Corporation for their kind cooperation.

References

Proceedings:

1. M. Fujita, Y. Kuroki, T. Ishida and T. Doi, A small humanoid robot SDR-4X for entertainment applications, *Int. Conf. on Advanced Intelligent Mechatronics (AIM)* (Kobe, JPN, 2003), pp. 938-943.

Authored book:

2. Robin R. Murphy, Introduction to AI ROBOTICS, The MIT Press, 2000.

Proceedings:

3. M. Fujita, R. Hasegawa, C. Gabriel, T. Takagi, J. Yokono and H. Shimomura, An Autonomous Robot that eats information via interaction with human and environment, *Int. Workshop on Robot-Human Interactive Communication (ROMAN)* (Bordeaux and Paris, FRN, 2001), pp.383-389.

Proceedings:

4. Fujita M., et. al., Physically and Emotionally grounded symbol acquisition for autonomous robots, *AAAI Fall Symposium: Emotional and Intelligent II* (Massachusetts, USA, 2001), pp. 43-46.

Proceedings:

5. R. Arkin, M. Fujita, T. Takagi and R. Hasegawa, Ethological Modeling and Architecture for an Entertainment Robot, *IEEE/RSJ Int. Conf. on Robotics and Automation (ICRA)* (Seoul, KOR, 2001).

Proceedings:

6. M. Fujita, Y. Kuroki, T. Ishida and T. Doi, Autonomous behavior control architecture of entertainment humanoid robot SDR-4X, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (IEEE Press, LA, USA, 2003), pp. 960-967.

Authored book:

7. Ekman, P. and Davidson, R. J., The nature of emotion, Oxford University Press, 1994.

Proceedings:

8. F. Tanaka, K. Noda, T. Sawada and M. Fujita, Associated Emotion and its Expression in an Entertainment Robot QRIO, *IFIP Int. Conf. Entertainment Computing (ICEC)* (Eindhoven, The Netherlands, 2004), pp. 499-504.

Proceedings:

9. Y. Hoshino, T. Takagi and M. Fujita, Behavior description and control using behavior module for personal robot, *IEEE/RSJ Int. Conf. on Robotics and Automation (ICRA)* (IEEE Press, Louisiana, USA, 2004), pp. 4165-4171.

Proceedings:

10. T. Sawada, T. Takagi and M. Fujita, Behavior Selection and Motion Modulation in Emotionally Grounded Architecture for QRIO SDR-4X, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (IEEE Press, Sendai, JPN, 2004), in press.

Proceedings:

11. Mark Humphrys, Action Selection methods using Reinforcement Learning, *Int. Conf. of Simulation of Adaptive Behavior (SAB)* (1996), pp. 135-144.

Proceedings:

12. K. Doya, K. Samejima, K. Katagiria and M. Kawato, Multiple model-based reinforcement learning,.Neural Computation, 2002, pp. 1347-1369.

Tsutomu Sawada received his M.S. and Ph.D. degrees from Science University of Tokyo, Japan, in 1998 and 2001, respectively. From 2001 he was researcher at the Digital Creatures Laboratory, Sony Corporation, and is currently working on development of behavior control architecture for entertainment robot SDR-4XII QRIO at Intelligent Systems Research Laboratory, Information Technologies Laboratories, Sony Corporation.
His research interests include reinforcement learning, design of sensory motor coordination and morphology, behavior control architecture and inferential system for human-machine interaction.

Tsuyoshi Takagi received a B.A in Mechanism from the Keio University, Tokyo, in 1994 and M.S. degree in mechanical engineering from the Keio University, Tokyo, in 1996. He joined Robot Entertainment project from 1998, and developed entertainment robot AIBO, which was started to sell in 1999. After the AIBO project, he worked on the development of behavior control architecture based on ethology at the Digital Creatures Laboratory, Sony Corporation, and is currently working for development of behavior control architecture for entertainment robot at Entertainment Robot Company, Sony Corporation. His research interests include agent architecture, autonomous behavior, evolutionary systems, cognitive science, social interaction, and learning.

Yukiko Hoshino received her M.S. and her Ph.D. degree in Mechano-Informatics from the University of Tokyo, Japan, in 1998 and 2001, respectively. From 2001, she was at the Digital Creatures Laboratory, Sony Corporation, and is currently at the Life Dynamics Laboratory Preparatory Office, Sony Corporation.

Yukiko Hoshino works in the human robot interaction and robot behavior system, and her recent work includes the development of behavior selection architecture and human robot interaction for QRIO SDR-4XII. Her research interests include human-robot interaction, embodiment and behavior coordination of the robot. Also, she received the 13th Best Paper Award from The Robotics Society of Japan, in 1999, for the work of full-body tactile sensor suit, and also received the 11th Young Investigator Excellence Award from the Robotics Society of Japan, in 1996.

Masahiro Fujita is a General Manager/Chief Researcher at Information Technologies Laboratories and a Research Director at Life Dynamics Laboratory Preparatory Office in Sony Corporation. He received a B.A. degree in Electronics and Communications from the Waseda University, Tokyo, in 1981, and joined Sony Corporation. He worked for development of a spread spectrum communication system, which was used for global positioning system in car navigation. From 1988, he became a graduate student of University of California, Irvine, and studied artificial neural network for visual perception. He received an M.S. degree in Electrical Engineering from the University of California, Irvine, in 1989. He started the Robot Entertainment project from 1993, and developed the entertainment robot AIBO, which started to sell in 1999. After the AIBO project, he has been in charge of the development of the cognitive component of a small humanoid robot QRIO. His research interests include computer vision, verbal and non-verbal interaction, language acquisition, emotional model of autonomous agents, and behavior control architecture.